Journal of Symbolic Computation 36 (2003) 235–269

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

On the complexity of equational problems in CNF☆

Reinhard Pichler*

Technische Universität Wien, Institut für Computersprachen, Abteilung für Anwendungen der Formalen Logik, Favoritenstrasse 9, E185-2, A-1040 Wien, Austria

Received 29 September 2000; accepted 24 April 2001

Abstract

Equational problems (i.e. first-order formulae with quantifier prefix $\exists^* \forall^*$, whose only predicate symbol is syntactic equality) are an important tool in many areas of automated deduction, e.g. restricting the set of ground instances of a clause via equational constraints allows the definition of stronger redundancy criteria and hence, in general, of more efficient theorem provers. Moreover, the inference rules themselves can be restricted via constraints. In automated model building, equational problems play an important role both in the definition of an appropriate model representation and in the evaluation of clauses in such models. Also, many problems in the area of logic programming can be reduced to equational problem solving. Finally, equational problems over a finite domain correspond to the evaluation of certain queries over relational databases.

The goal of this work is a complexity analysis of the satisfiability problem of equational problems. The main results will be a proof of the NP-completeness (and, in particular, the NP-membership) of equational problems in CNF over an infinite domain and of the Σ_2^p-completeness in the case of CNF over a finite domain.

Keywords: Equational reasoning; Complexity; NP-completeness; Polynomial hierarchy

1. Introduction

Equational problems (i.e. first-order formulae with quantifier prefix $\exists^* \forall^*$, whose only predicate symbol is syntactic equality) are an important tool in many areas of automated deduction.

Restricting the set of ground instances of a clause via equational constraints allows the definition of stronger redundancy criteria and hence, in general, of more efficient theorem provers: in Caferra and Zabel (1991), the c-distautology rule allows the deletion

☆ Preliminary versions of the results contained in this paper were presented at CADE-16 and FTP 2000 (cf. Pichler, 1999 and Pichler, 2000a, respectively).

* Tel.: +43-5170721851.

E-mail address: reini@logic.at (R. Pichler).

0014-5793/03/$ - see front matter
doi:10.1016/S0747-7171(03)00029-4

of those ground instances of a clause, which are tautological. To this end, the instances of a clause $C = P(\vec{s}) \vee \neg P(\vec{t}) \vee C'$ are restricted via the constraint $\vec{s} \neq \vec{t}$. Likewise, the c-dissubsumption rule allows the deletion of those ground instances of a clause which are subsumed by another clause. Moreover, in Caferra and Peltier (1995), semantic c-resolution is defined, where the inference rules themselves are restricted via equational constraints.

In automated model building, equational problems play a crucial role, e.g. in Caferra and Zabel (1991), models are represented by unit clauses whose set of ground instances may be restricted via equational constraints. In Fermüller and Leitsch (1996), models are represented by atom sets $\mathcal{A} = \{P_1(\vec{t}_1), \ldots, P_n(\vec{t}_n)\}$ with the intended meaning that a ground atom $P(\vec{r})$ evaluates to "true", iff it is an instance of some atom $P_i(\vec{t}_i) \in \mathcal{A}$. The computation of the truth value of a non-ground atom $P(\vec{s})$ in such a model can be easily reduced to the satisfiability problem of an equational problem, namely: let $\vec{x}$ denote the (vector of) variables in $P(\vec{s})$ and let $\vec{y}$ be another vector of variables, such that $\vec{x}$ and $\vec{y}$ have no variables in common and all variables occurring in any of the atoms $P_i(\vec{t}_i) \in \mathcal{A}$ are contained in $\vec{y}$. Then the condition, that every ground instance of $P(\vec{s})$ is also a ground instance of one of the atoms $P_i(\vec{t}_i) \in \mathcal{A}$, corresponds to the validity of the formula $(\forall\vec{x})(\exists\vec{y}) \bigvee_{i=1}^{n} P(\vec{s}) = P_i(\vec{t}_i)$, which is equivalent to $(\forall\vec{x})(\exists\vec{y}) \bigvee_{P_i=P} (\vec{s} = \vec{t}_i)$. In other words, the atom $P(\vec{s})$ evaluates to "false", iff the equational problem $\mathcal{P} \equiv (\exists\vec{x})(\forall\vec{y}) \bigwedge_{P_i=P} (\vec{s} \neq \vec{t}_i)$ is satisfiable.

In Lassez and Marriott (1987), implicit generalizations are studied as a formal basis of machine learning from counter-examples. For testing the emptiness of implicit generalizations, equational problems similar to the above-mentioned evaluation of atoms arise. Likewise, in functional programming, the problem of checking whether a function defined by case is completely defined, can be reduced to equational problems of this kind (cf. Lassez et al., 1991). In the area of logic programming, equational problems may be applied in various ways. In particular, the definition of an appropriate semantics of negation can be reduced to equational problem solving (cf. Lassez and Marriott, 1987; Lugiez, 1989; Sato and Motoyoshi, 1991).

A Boolean conjunctive query Q on a relational database DB over some (finite) universe U is a construct of the form "$\leftarrow P_1(X_{11}, \ldots, X_{1s_1}), \ldots, P_k(X_{k1}, \ldots, X_{ks_k})$", where every P_i is a relation symbol from the database DB. The variables X_{ij} may take values from the universe U. Of course, these variables are, in general, not distinct. Intuitively, Q evaluates to "true" on DB, if the variables X_{ij} can be simultaneously replaced by values u_{ij} from the universe U, such that the resulting ground atoms $P_1(u_{11}, \ldots, u_{1s_1}), \ldots, P_k(u_{k1}, \ldots, u_{ks_k})$ are contained in the database DB. The latter condition corresponds to the satisfiability of the equational problem $(\exists\vec{X})[\bigwedge_{i=1}^{k} \bigvee_{P_i(u_{i1},\ldots,u_{is_i}) \in DB} (X_{i1} = u_{i1} \wedge \cdots \wedge X_{is_i} = u_{is_i})]$, where $\vec{X}$ denotes the set of all variables X_{ij}. For details on database theory, see e.g. Abiteboul et al. (1995) or Ullman (1989).

In many of the above-mentioned applications, testing the satisfiability of an equational problem is even more important than actually computing the solutions. The usefulness of equational problems mainly comes from their balance between expressive power and computational complexity. In particular, recall from Kunen (1987) and Vorobyov (1996) that if we did not impose any restrictions on the quantifier prefix, then the satisfiability

problem would be PSPACE-complete (in the case of a finite domain) or even non-elementary recursive (for an infinite domain), respectively. The goal of this work is an investigation of the inherent complexity of the satisfiability problem of equational problems (i.e. where the quantifier prefix is of the form $\exists^* \forall^*$). Our main results will be a proof of the NP-completeness (and, in particular, the NP-membership) for equational problems in CNF over an infinite domain and of the Σ_2^p-completeness in the case of CNF over a finite domain. In summary, we get the following results:

	Finite domain *D*		**Infinite domain**
	$\lvert D\rvert = 2$	$\lvert D\rvert \geq 3$	
CNF	NP-complete	$\boldsymbol{\Sigma_2^p}$**-complete**	**NP-complete**
DNF	Σ_2^p-complete	Σ_2^p-complete	Σ_2^p-hard

This work is organized as follows: after briefly recalling some basic definitions in Section 2, we shall consider in Section 3 the special case of purely existentially quantified equational problems in CNF. In Sections 4 and 5, we shall provide a thorough complexity analysis of equational problems in the case of a finite domain and an infinite domain, respectively. In Section 6, these complexity considerations are applied to the construction of more efficient satisfiability algorithms of equational problems. Finally, in Section 7, we shall summarize the main results of this work and point out some directions for future research. For a quick reference, the transformation rules of Comon and Lescanne (1989) are recalled in the Appendix.

2. Preliminaries

2.1. Equational problems

Equational problems are first-order formulae of the form $\exists\vec{w}\forall\vec{y}\mathcal{P}(\vec{w},\vec{x},\vec{y})$, such that $\mathcal{P}(\vec{w},\vec{x},\vec{y})$ is a quantifier-free formula with variables in $\vec{w}$, $\vec{x}$ and $\vec{y}$, where syntactic equality "$=$" is the only predicate symbol. A disequation $s \neq t$ is a short-hand notation for a negated equation $\neg(s = t)$. The trivially true problem is denoted by $\top$ and the trivially false one by $\bot$.

In this paper, every equational problem $\mathcal{P}$ is considered over some fixed finite signature Σ consisting of constant symbols and possibly function symbols. We assume that Σ contains at least one constant symbol. Moreover, the domain D over which the terms and the variables of $\mathcal{P}$ are interpreted is the Herbrand universe over the signature Σ, i.e. the algebra of ground terms that can be constructed from the symbols in Σ. Clearly, this domain is infinite, iff Σ contains at least one (proper) function symbol. Throughout this paper, we only consider the case of a non-trivial domain, i.e. a domain with at least two elements. In other words, the signature Σ also has at least two elements.

An interpretation over D is given through a D-ground substitution σ, whose domain coincides with the free variables of the equational problem. The trivial problem $\top$ evaluates to "true" in every interpretation. Likewise, $\bot$ always evaluates to "false". A single equation $s = t$ is validated by a ground substitution σ, iff $s\sigma$ and $t\sigma$ are syntactically identical.

Note that ground terms are interpreted "by themselves", so to speak. Analogously to the usual treatment of equational problems in the literature, we do not distinguish between a constant *symbol* c in the signature Σ and the constant (i.e. *element*) c in the domain D. The connectives $\wedge$, $\vee$, $\neg$, $\exists$ and $\forall$ are interpreted as usual. A ground substitution σ which validates a problem $\mathcal{P}$ is called a *solution* of $\mathcal{P}$. An equational problem is *satisfiable*, iff it has at least one solution.

As far as the satisfiability of an equational problem is concerned, there is no difference between free variables and existentially quantified ones, i.e. $\exists\vec{w}\forall\vec{y}\mathcal{P}(\vec{w},\vec{x},\vec{y})$ is satisfiable, iff $\exists\vec{x}\exists\vec{w}\forall\vec{y}\mathcal{P}(\vec{w},\vec{x},\vec{y})$ is. Without loss of generality, we therefore only consider equational problems without free variables. In analogy with Comon and Lescanne (1989), universally quantified variables will be referred to as *parameters*.

For an arbitrary but fixed domain D with signature Σ we study (various variants of) the following decision problem in this paper:

Equational Problem Satisfiability

Input: A first-order formula $\exists\vec{x}\forall\vec{y}\mathcal{P}(\vec{x},\vec{y})$ in prenex normal form over the domain D, whose only predicate symbol is the syntactic equality "=".

Question: Is the formula $\exists\vec{x}\forall\vec{y}\mathcal{P}(\vec{x},\vec{y})$ satisfiable?

In the above definition we consider the domain D as arbitrary but fixed. So, in principle, we have to deal with a collection of decision problems, which are in a sense "parameterized" by D. Note however that we get exactly the same complexity results, if the domain D is considered as part of the input. In particular, the NP-membership proof in Section 2.1 for the case of a fixed domain can be taken over literally to the case where the domain is part of the input.

In order to distinguish between syntactical identity and the equivalence of two equational problems, we use the notation "$\equiv$" and "$\approx$", respectively, i.e. $\mathcal{P}\equiv\mathcal{Q}$ means that the two equational problems $\mathcal{P}$ and $\mathcal{Q}$ are *syntactically identical*, while $\mathcal{P}\approx\mathcal{Q}$ means that the two problems are *equivalent* (i.e. they have the same set of solutions). If an equational formula $\mathcal{P}$ contains no variables (i.e. it is made up from ground equations and disequations), then it either evaluates to "true" in every interpretation or it evaluates to "false" in every interpretation. Hence, we either have $\mathcal{P}\approx\top$ (i.e. "$\mathcal{P}$ is trivially true") or $\mathcal{P}\approx\bot$ (i.e. "$\mathcal{P}$ is trivially false").

We shall sometimes use term tuples as a shorthand notation for a conjunction of equations or a disjunction of disequations, respectively, i.e. for term tuples $\vec{s}=(s_1,\ldots,s_k)$ and $\vec{t}=(t_1,\ldots,t_k)$, we shall abbreviate "$s_1=t_1\wedge\cdots\wedge s_k=t_k$" and "$s_1\neq t_1\vee\cdots\vee s_k\neq t_k$" to "$\vec{s}=\vec{t}$" and "$\vec{s}\neq\vec{t}$", respectively. Moreover, we shall use a vector $\vec{x}=(x_1,\ldots,x_k)$ of variables either to denote a tuple of variables (which can be used as an argument of an equation or disequation in the way described above) or to denote a set of variables. No ambiguities will arise from this, since the meaning will always be clear from the context.

2.2. The transformation rules of Comon and Lescanne (1989)

In Comon and Lescanne (1989), a rule system is provided which terminates on every equational problem and which transforms the original problem into an equivalent one in the so-called "definition with constraints form", which allows to determine immediately

the satisfiability. Below, those rules of Comon and Lescanne (1989) are recalled, which are cited frequently in this paper, i.e. the replacement rules R_1, R_2, the universality of parameter rules U_2, U_4, U_5 and the explosion rule E. Note that many more rules of Comon and Lescanne (1989) (like the decomposition rule, the clash rule, the occur check, etc.), which are not mentioned explicitly here, are "hidden" in the unification steps. A list of the relevant transformation rules of Comon and Lescanne (1989) is given in the Appendix.

(R_1) $z = t \wedge P \to z = t \wedge P(z \leftarrow t)$
(R_2) $z \neq t \vee P \to z \neq t \vee P(z \leftarrow t)$
(U_2) $(\forall \vec{y})[P \wedge (y \neq t \vee R)] \to (\forall \vec{y})[P \wedge R(y \leftarrow t)]$
if the following conditions hold:

1. $y \in \vec{y}$,
2. $y \notin \mathrm{Var}(t)$.

(E) $(\forall \vec{y})P \to \bigvee_{f \in \Sigma} (\exists \vec{w})(\forall \vec{y})[P \wedge s = f(w_1, \ldots, w_{\alpha(f)})]$
if the following conditions hold:

1. Each f is a (constant or function) symbol from the signature Σ with arity $\alpha(f) \geq 0$,
2. the w_i's are fresh, pairwise distinct variables,
3. s is an argument of an equation or disequation in P and s contains no parameter.

The following rule is only correct in the case of an infinite domain:

(U_4) $(\forall \vec{y})[P \wedge (z_1 = u_1 \vee \cdots \vee z_n = u_n \vee R)] \to (\forall \vec{y})[P \wedge R]$
if the following conditions hold:

1. Every z_i is a variable syntactically different from u_i,
2. every equation $z_i = u_i$ contains at least one parameter from $\vec{y}$,
3. R contains no parameter from $\vec{y}$.

The following rule can only be applied in case of a finite domain:

(U_5) $(\forall \vec{y})[P \wedge Q] \to (\forall \vec{y})[P \wedge Q(y \leftarrow a_1) \wedge \cdots \wedge Q(y \leftarrow a_K)]$ if the domain D is of the form $D = \{a_1, \ldots, a_K\}$.

The correctness of the rule R_1 is obvious. The rule R_2 essentially follows from the equivalence $[A \vee B] \approx [(A \wedge \neg B) \vee B]$, which holds for any logical formulae A and B. The correctness of the U_2-rule is then also easy to see: of course, we may shift the universal quantifiers inside the $\wedge$, i.e. $(\forall \vec{y})[P \wedge (y \neq t \vee R)] \approx [(\forall \vec{y})P] \wedge (\forall \vec{y})[y \neq t \vee R]$. Then the formula on the right-hand side of the U_2-rule is obtained by applying the rule R_2 and the fact that the disequation $(\forall \vec{y})y \neq t$ is false over any non-trivial domain.

The explosion rule E (and, analogously, the U_5-rule) is sometimes also referred to as the domain closure axiom. Its idea is the following: let H be the Herbrand universe of terms over some finite signature Σ. Then every ground term $t \in H$ has one of the symbols in Σ as its leading symbol. Hence the formula $\bigvee_{f \in \Sigma} (\exists \vec{w})[s = f(w_1, \ldots, w_{\alpha(f)})]$ is clearly valid for any term s. Note that the third condition in the definition of the rule E is only needed for the termination of the rule system in Comon and Lescanne (1989) and not for the correctness of the rule itself.

Finally, the rule U_4 is due to the so-called *independence of inequations* of Colmerauer (1984) (see also Lemma 1 in Comon and Lescanne, 1989), namely: every purely existentially quantified conjunction of disequations over an infinite domain has at least one solution, iff each of the conjuncts has a solution. The latter condition is always fulfilled unless one of the conjuncts is a trivial disequation of the form $t \neq t$. Then the correctness of the U_4-rule is mainly due to the fact that the subformula $(z_1 = u_1 \vee \cdots \vee z_n = u_n)$ cannot be true for all values of the variables in $\vec{y}$, since the negation $(\exists \vec{y})(z_1 \neq u_1 \wedge \cdots \wedge z_n \neq u_n)$ is a conjunction of non-trivial disequations, which is satisfiable by the independence of inequations.

2.3. Unification problems

If an equational problem is a parameter-free conjunction of equations, then we are back to the familiar case of *unification problems*. Let $\mathcal{P} \equiv s_1 = t_1 \wedge \cdots \wedge s_n = t_n$ be a quantifier-free conjunction of equations and suppose that the set $\mathcal{S}$ of solutions of $\mathcal{P}$ is non-empty. Then $\mathcal{S}$ can be represented by a single (in general, non-ground) substitution μ, which is called the mgu (=most general unifier) of $\mathcal{P}$, i.e. for every solution σ of $\mathcal{P}$, there exists a substitution η, such that σ is the composition of μ and η (which we denote by $\sigma = \mu \circ \eta$ or simply $\sigma = \mu\eta$). Recall that the mgu is unique up to variable renaming.

We write substitutions in the form $\mu = \{x_1 \leftarrow r_1, \ldots, x_n \leftarrow r_n\}$, where the set of variables $\{x_1, \ldots, x_n\}$ is called the *domain* of μ, which is denoted by $\mathrm{dom}(\mu)$. The set of terms $\{r_1, \ldots, r_n\}$ is referred to as the *range* of μ, which we denote by $\mathrm{rg}(\mu)$. If λ and μ are two substitutions, such that the domain and range of λ have no variables in common with the domain and range of μ, then we shall sometimes also write $\lambda \cup \mu$ to denote the composition $\lambda \circ \mu$ of these substitutions. Finally, if μ is the mgu of a quantifier-free conjunction of equations $\mathcal{P} \equiv s_1 = t_1 \wedge \cdots \wedge s_n = t_n$, then we may assume without loss of generality that μ contains no new variables, i.e. $\mathrm{dom}(\mu)$ and the variables occurring in $\mathrm{rg}(\mu)$ also occur in $\mathcal{P}$.

In Baader and Siekmann (1994), the efficiency of several unification algorithms is analysed. In particular, it is shown, that the original unification algorithm of Robinson (1965), where terms are represented as strings of symbols, has exponential time and space complexity. However, by using more sophisticated data structures like directed acyclic graphs, this kind of exponential blow-up can be avoided. In fact, even linear time suffices (cf. Martelli and Montanari, 1982).

2.4. Quantified Boolean formulae

Our Σ_2^p-hardness proofs in Section 4 will be based on the 3-QSAT$_2$ problem (=*quantified satisfiability with two alternating blocks of quantifiers*), which is a well-known Σ_2^p-complete problem (cf. Stockmeyer, 1976). It is defined as follows:

3-QSAT$_2$

Input: A triple (P, R, E), such that $P = \{p_1, \ldots, p_k\}$ and $R = \{r_1, \ldots, r_l\}$ are disjoint sets of propositional variables and $E = (l_{11} \wedge l_{12} \wedge l_{13}) \vee \cdots \vee (l_{n1} \wedge l_{n2} \wedge l_{n3})$ is a Boolean formula with propositional variables in $P \cup R$.

Question: Is the quantified Boolean sentence $\exists(p_1, \ldots, p_k)\, \forall(r_1, \ldots, r_l)\, E$ satisfiable?

The literals $l_{\alpha\beta}$ in E are unnegated or negated propositional variables from $P \cup R$. A literal $l_{\alpha\beta}$ of the form p_i or $\neg p_i$ for some $p_i \in P$ will be referred to as a "literal over P". Likewise, a "literal over R" is of the form r_i or $\neg r_i$ for some $r_i \in R$.

In this paper, it is convenient to restrict the form of the Boolean formula E in several ways: we assume that no conjunction $C_i = l_{i1} \wedge l_{i2} \wedge l_{i3}$ contains a pair of complementary literals (since otherwise this conjunction will never evaluate to "true"). Moreover, we assume that every conjunction C_i contains at least one literal over R (since otherwise it is trivial to define a truth assignment $\mathcal{I}$ on P such that C_i evaluates to "true" in $\mathcal{I}$ and, hence, in every extension $\mathcal{J}$ of $\mathcal{I}$ to $P \cup R$). Finally we may of course arrange the literals of each conjunction C_i in such a way that the literals over R stand in front of the literals over P.

If we are not interested in the precise truth value of some propositional variables in $E = (l_{11} \wedge l_{12} \wedge l_{13}) \vee \cdots \vee (l_{n1} \wedge l_{n2} \wedge l_{n3})$, then we shall consider *partial* truth assignments $\mathcal{J}$ on the propositional variables occurring in E. By "$\mathcal{J}(v) =$ undefined" we denote that $\mathcal{J}$ assigns no truth value to v. To this end, we need the following generalization of the evaluation of Boolean formulae: we say that E evaluates to "false" in a partial assignment $\mathcal{J}$, iff E evaluates to "false" in every complete extension $\mathcal{J}'$ of $\mathcal{J}$. In particular, if $\mathcal{J}$ is a partial assignment such that in every conjunction $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ of E there is at least one literal $l_{\alpha\beta}$ that evaluates to "false" in $\mathcal{J}$, then E evaluates to "false" in $\mathcal{J}$.

3. Parameterless CNF

The first goal of the algorithm of Comon and Lescanne (1989) is the elimination of all universally quantified variables. Together with the restriction to CNF, we get the following form:

Definition (Parameterless CNF). An equational problem $\mathcal{P}$ is said to be in *parameterless CNF*, iff it is in CNF and contains no universally quantified variables, i.e. $\mathcal{P} \equiv (\exists \vec{x})[(e_{11} \vee \cdots \vee e_{1k_1} \vee d_{11} \vee \cdots \vee d_{1l_1}) \wedge \cdots \wedge (e_{n1} \vee \cdots \vee e_{nk_n} \vee d_{n1} \vee \cdots \vee d_{nl_n})]$, such that the e_{ij}'s are equations and the d_{ij}'s are disequations.

Testing the satisfiability of equational problems in parameterless CNF can be easily shown to be NP-complete for any non-trivial domain. At the heart of the NP-membership proof in the case of an infinite domain is the polynomial time test for the satisfiability of parameterless conjunctions of equations and disequations given in Lemma 3.1 below. This test will also play an essential role in Section 5, when we prove the NP-membership of equational problems in CNF over an infinite domain, even if these problems do contain parameters.

Lemma 3.1 (Parameterless Conjunctions). *Let $\mathcal{P} \equiv (\exists \vec{x})[e_1 \wedge \cdots \wedge e_k \wedge d_1 \wedge \cdots \wedge d_l]$ be a conjunction of equations e_i and disequations d_j over some infinite domain D. Then the satisfiability of $\mathcal{P}$ can be tested as follows:*

- *Case 1: If $e_1 \wedge \cdots \wedge e_k$ is unsatisfiable, then $\mathcal{P}$ is also unsatisfiable.*
- *Case 2: Let $e_1 \wedge \cdots \wedge e_k$ be satisfiable with* mgu ϑ. *Then $\mathcal{P}$ is satisfiable, iff $d_1\vartheta \wedge \cdots \wedge d_l\vartheta$ contains no trivial disequation of the form $t \neq t$.*

Proof. Case 1 is trivial. For Case 2, let $\vartheta = \{x_{i_1} \leftarrow s_1, \ldots, x_{i_\alpha} \leftarrow s_\alpha\}$ denote the mgu of the equations $e_1 \wedge \cdots \wedge e_k$. Note that the variables x_{i_j} are pairwise distinct and do not occur in the range of ϑ.

By the definition of the mgu, the conjunctions $e_1 \wedge \cdots \wedge e_k$ and $x_{i_1} = s_1 \wedge \cdots \wedge x_{i_\alpha} = s_\alpha$ are equivalent. Moreover, by multiple applications of the R_2-rule recalled in Section 2.2, ϑ may be applied to the disequations. Thus $\mathcal{P} \approx (\exists \vec{x})[x_{i_1} = s_1 \wedge \cdots \wedge x_{i_\alpha} = s_\alpha \wedge d_1\vartheta \wedge \cdots \wedge d_l\vartheta]$ holds. But then, since all variables $x_{i_1}, \ldots, x_{i_\alpha}$ occur only once, the equations may be eliminated by the CR_2-rule of Comon and Lescanne (1989), i.e. $\mathcal{P} \approx \mathcal{P}' \equiv (\exists \vec{x})[d_1\vartheta \wedge \cdots \wedge d_l\vartheta]$. By the independence of inequations recalled in Section 2.2, any conjunction of non-trivial disequations over an infinite domain has at least one solution. Therefore, $\mathcal{P}'$ (and, hence, also $\mathcal{P}$) is indeed satisfiable, iff $\mathcal{P}'$ contains no disequation of the form $t \neq t$. □

We thus get the following theorem:

Theorem 3.1 (NP-completeness of Parameterless CNF). *Let D be an arbitrary domain (i.e. finite or infinite) with at least two elements. Then the satisfiability problem for equational problems in parameterless CNF over D is NP-complete.*

Proof (Sketch). Let $\mathcal{P} \equiv (\exists \vec{x})[(e_{11} \vee \cdots \vee e_{1k_1} \vee d_{11} \vee \cdots \vee d_{1l_1}) \wedge \cdots \wedge [(e_{n1} \vee \cdots \vee e_{nk_n} \vee d_{n1} \vee \cdots \vee d_{nl_n})]$ be an equational problem in parameterless CNF. For proving the *NP-membership*, we have to treat the cases of a finite and an infinite domain separately, i.e. if D is finite, then the satisfiability of $\mathcal{P}$ can be tested by guessing values of $\vec{x}$ in D and checking that the resulting variable-free equational formula is trivially true. On the other hand, if D is infinite, then we make use of Lemma 3.1, namely: from every clause in $\mathcal{P}$, guess a disjunct $\mathcal{D}_i$, i.e. for every $i \in \{1, \ldots, n\}$, $\mathcal{D}_i$ is either an equation $e_{i\alpha_i}$ or a disequation $d_{i\alpha_i}$. Then, by Lemma 3.1, we can check in polynomial time, whether the equational problem $(\exists \vec{x})[\mathcal{D}_1 \wedge \cdots \wedge \mathcal{D}_n]$ is satisfiable. The *NP-hardness* can be shown by the obvious reduction from the 3-SAT problem, i.e.

3-SAT

Input: A Boolean formula $E = (l_{11} \vee l_{12} \vee l_{13}) \wedge \cdots \wedge (l_{n1} \vee l_{n2} \vee l_{n3})$ with propositional variables in $P = \{p_1, \ldots, p_k\}$.

Question: Is E satisfiable?

Now let a be an arbitrary constant in D. Then we define the equational problem $\mathcal{P} \equiv \exists \vec{x}\,[(l'_{11} \vee l'_{12} \vee l'_{13}) \wedge \cdots \wedge (l'_{n1} \vee l'_{n2} \vee l'_{n3})]$ in parameterless CNF over D with $\vec{x} = (x_1, \ldots, x_k)$, such that the literals l'_{ij} in $\mathcal{P}$ are defined as follows:

$$l'_{ij} = \begin{cases} x_\gamma = a & \text{if } l_{ij} \text{ is a positive literal } p_\gamma \\ x_\gamma \neq a & \text{if } l_{ij} \text{ is a negative literal } \neg p_\gamma. \end{cases}$$

In other words, the constant $a \in D$ is used to encode the truth value "true" and every literal of the form p_γ or $\neg p_\gamma$ in E is encoded by the literal $x_\gamma = a$ or $x_\gamma \neq a$, respectively, in $\mathcal{P}$. Clearly, this reduction is feasible in polynomial time. Moreover, its correctness is trivial. □

4. Equational problems over a finite domain

In this section, we provide a complexity analysis of the satisfiability problem of equational problems over a finite domain. For equational problems in arbitrary form or in DNF, the Σ_2^p-completeness follows immediately from the Σ_2^p-completeness of the 3-QSAT$_2$ problem recalled in Section 2.4. We therefore have:

Theorem 4.1 (Arbitrary Form or DNF Over a Finite Domain). *Let D be a finite domain with at least two elements. Then the satisfiability problem for equational problems in arbitrary form or in DNF over D is Σ_2^p-complete.*

Proof (Sketch). The *Σ_2^p-membership* is trivial, i.e. guess values for the existentially quantified variables and check the satisfiability of the resulting formula by means of an NP-oracle. The *Σ_2^p-hardness* proof via a reduction from the 3-QSAT$_2$ problem follows exactly the same pattern as the NP-hardness proof in Theorem 3.1, i.e. let (P, R, E) be an instance of the 3-QSAT$_2$ problem and let a be an arbitrary constant in D. Then we define the equational problem $\mathcal{P} \equiv \exists\vec{x}\forall\vec{y}\,[C_1 \wedge \cdots \wedge C_n]$ in DNF over D in such a way that every literal of the form p_γ or $\neg p_\gamma$ in E is encoded by the literal $x_\gamma = a$ or $x_\gamma \neq a$, respectively, in $\mathcal{P}$. Likewise, $y_\beta = a$ and $y_\beta \neq a$ are used to encode literals of the form r_β or $\neg r_\beta$, respectively. Again, this reduction can be clearly done in polynomial time and its correctness is trivial. □

The remainder of this section deals with the complexity of equational problems in CNF. The idea of the Σ_2^p-completeness proof is illustrated in Theorem 4.2 below, where we consider the case of a domain with exactly three elements. The extension to a K-element domain for some arbitrary $K \geq 3$ will then be straightforward.

Theorem 4.2 (CNF Over a Three-Element Domain). *Let D be a domain with three elements. Then the satisfiability problem for equational problems in CNF over D is Σ_2^p-complete.*

Proof. The *Σ_2^p-membership* is clear by Theorem 4.1. For the *Σ_2^p-hardness* we consider the following reduction of the 3-QSAT$_2$ problem to the satisfiability problem of equational problems: let $\mathcal{Q} = (P, R, E)$ be an instance of the 3-QSAT$_2$ problem, where $P = \{p_1, \ldots, p_k\}$ and $R = \{r_1, \ldots, r_l\}$ are sets of propositional variables and $E = (l_{11} \wedge l_{12} \wedge l_{13}) \vee \cdots \vee (l_{n1} \wedge l_{n2} \wedge l_{n3})$ is a Boolean formula with propositional variables in $P \cup R$. Then we define the equational problem $\mathcal{P} \equiv \exists\vec{x}\forall\vec{y}\forall\vec{z}\,[C_0 \wedge \mathcal{C}]$ in CNF over $D = \{a, b, c\}$ as follows:

Variables:

- "$\vec{x}$": For every literal $l_{\alpha\beta}$ over P (i.e. $l_{\alpha\beta}$ is of the form p_i or $\neg p_i$ for some $p_i \in P$), there is a first-order variable $x_{\alpha\beta}$ in $\vec{x}$.
- "$\vec{y}$": For every literal $l_{\alpha\beta}$ over R, there is a first-order variable $y_{\alpha\beta}$ in $\vec{y}$.
- "$\vec{z}$": For every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R with $l_{\alpha\beta} \in R$ and $l_{\gamma\delta} = \neg l_{\alpha\beta}$, there are three first-order variables z_1, z_2 and z_3 in $\vec{z}$.

All of the variables in $\vec{x}$, $\vec{y}$ and $\vec{z}$ mentioned above are assumed to be pairwise distinct[1]. Note that the division of the universally quantified variables into $\vec{y}$ and $\vec{z}$ was only done for the sake of better readability. We still have a quantifier prefix of the form $\exists^* \forall^*$.

Clause C_0: There is a "big clause" C_0, which contains some information about the conjunctions of E and the complementary literals over R that occur in E. The clause C_0 consists of the following disjuncts:

- *Case 1*: If all of the literals $l_{\alpha 1}, l_{\alpha 2}$ and $l_{\alpha 3}$ in the αth conjunction of E are literals over R (i.e. they are of the form r_i or $\neg r_i$ for some $r_i \in R$), then C_0 contains the three disjuncts $y_{\alpha 1} = y_{\alpha 2}$, $y_{\alpha 1} = y_{\alpha 3}$ and $y_{\alpha 2} = y_{\alpha 3}$.
- *Case 2*: If in the αth conjunction of E, there are two literals $l_{\alpha 1}$ and $l_{\alpha 2}$ over R and one literal $l_{\alpha 3}$ over P, then C_0 contains the three disjuncts $y_{\alpha 1} = y_{\alpha 2}$, $y_{\alpha 1} = x_{\alpha 3}$ and $y_{\alpha 2} = x_{\alpha 3}$.
- *Case 3*: Finally, if in the αth conjunction of E there is one literal $l_{\alpha 1}$ over R and two literals $l_{\alpha 2}$ and $l_{\alpha 3}$ over P, then C_0 contains the two disjuncts $y_{\alpha 1} = x_{\alpha 2}$ and $y_{\alpha 1} = x_{\alpha 3}$.

Moreover, for every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R, where $l_{\alpha\beta} \in R$ and $l_{\gamma\delta} = \neg l_{\alpha\beta}$ hold, C_0 contains the following six disjuncts:

$$z_1 = z_2 \quad z_1 = z_3 \quad z_2 = z_3 \quad z_1 = a \quad z_2 = y_{\alpha\beta} \quad z_3 = y_{\gamma\delta}.$$

No further disjuncts are contained in C_0.

Before we give a definition of the conjunction $\mathcal{C}$ of clauses, let us briefly explain the idea of the big clause C_0 defined above: actually, we shall primarily have to deal with the negated form $\neg C_0$, which is a conjunction of disequations over the three-element domain $D = \{a, b, c\}$. Hence, $\neg C_0$ can be considered as the encoding of a 3-colourability problem of a graph G whose vertices correspond to the variables in $\neg C_0$ and whose edges correspond to the disequations in $\neg C_0$. By the correspondence between the literals $l_{\alpha\beta}$ in the Boolean formula E and the variables in $\vec{x} \cup \vec{y}$, the graph G contains a triangle with labels $\{y_{\alpha 1}, y_{\alpha 2}, y_{\alpha 3}\}$ or $\{y_{\alpha 1}, y_{\alpha 2}, x_{\alpha 3}\}$ or $\{y_{\alpha 1}, x_{\alpha 2}, x_{\alpha 3}\}$ for each conjunction $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ in the Boolean formula E. Only the edge between $x_{\alpha 2}$ and $x_{\alpha 3}$ is omitted in the third case. At any rate, there is a one-to-one correspondence between the vertices of these triangles and the literals in E. As far as the truth assignments for the literals in E are concerned, we are mainly interested in those assignments $\mathcal{J}$ where E has the truth value "false", i.e. in each conjunction $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ we want at least one literal to evaluate to "false". As for the colourings of the graph G with the colours $\{a, b, c\}$ (or, equivalently, the ground substitutions with domain $\vec{x} \cup \vec{y}$ and range $\{a, b, c\}$), the truth value "false" will be encoded by the element $a \in D$.

Note that the six disequations $z_1 \neq z_2, z_1 \neq z_3, z_2 \neq z_3, z_1 \neq a, z_2 \neq y_{\alpha\beta}$ and $z_3 \neq y_{\gamma\delta}$ in the negated clause $\neg C_0$ correspond to the subgraph depicted in Fig. 1. It can be easily

[1] Of course, for each pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals, we need a separate collection of variables z_1, z_2 and z_3. Hence, strictly speaking, we should refer to these variables in $\vec{z}$ as $z_{(\alpha\beta,\gamma\delta,1)}$, $z_{(\alpha\beta,\gamma\delta,2)}$ and $z_{(\alpha\beta,\gamma\delta,3)}$ (or something like this). However, for the sake of better readability, we have omitted this multiple indexing. The complementary literals $l_{\alpha\beta}$ and $l_{\gamma\delta}$ corresponding to some variables z_1, z_2 and z_3 in $\vec{z}$ will always be clear from the context.

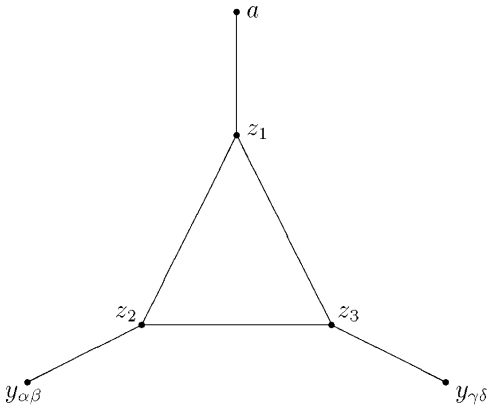

Fig. 1. Graph corresponding to disequations with the variables z_1, z_2 and z_3.

checked, that this gadget has a valid 3*-colouring, iff at least one of the vertices with label* $y_{\alpha\beta}$ *or* $y_{\gamma\delta}$ *is assigned a colour different from* a*. By the above-mentioned correspondence between truth values of the literals in* E *and instantiations of the variables in* $\vec{x} \cup \vec{y}$*, this means that the gadget in Fig.* 1 *has a valid* 3*-colouring, iff we do not assign the truth value "false" to both literals* $l_{\alpha\beta}$ *and* $l_{\gamma\delta}$ *in* E*.*

Clauses in $\mathcal{C}$: $\mathcal{C}$ is a conjunction of clauses with variables from $\vec{x}$ only. It consists of the following clauses:

- For every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over P, where $l_{\alpha\beta} \in P$ and $l_{\gamma\delta} = \neg l_{\alpha\beta}$ hold, $\mathcal{C}$ contains the following two clauses: $x_{\alpha\beta} = a \vee x_{\gamma\delta} = a$ and $x_{\alpha\beta} \neq a \vee x_{\gamma\delta} \neq a$.
- Moreover, for every conjunction in E with two literals $l_{\alpha 2}$ and $l_{\alpha 3}$ over P, the set $\mathcal{C}$ contains the clause $x_{\alpha 2} \neq x_{\alpha 3} \vee x_{\alpha 2} = a \vee x_{\alpha 3} = a$.

No further clauses are in $\mathcal{C}$.

Again we pause for a moment, in order to illustrate the idea of this definition. Note that the conjunction $\mathcal{C}$ *of clauses contains only variables from* $\vec{x}$*. Moreover, recall that the clause* $\neg C_0$ *defined above corresponds to the* 3*-colourability problem of a graph* G *and that the element* $a \in D$ *is used to encode the truth value "false" of the corresponding literals in* E*. Now the purpose of* $\mathcal{C}$ *is to restrict the choice of colours that we may possibly assign to the vertices in* G *with label* $x_{\alpha\beta} \in \vec{x}$*. In particular, for a pair of complementary literals* $\{l_{\alpha\beta}, l_{\gamma\delta}\}$ *in* E*, the clauses* $x_{\alpha\beta} = a \vee x_{\gamma\delta} = a$ *and* $x_{\alpha\beta} \neq a \vee x_{\gamma\delta} \neq a$ *ensure that exactly one of the vertices in* G *with label* $x_{\alpha\beta}$ *and* $x_{\gamma\delta}$*, respectively, is assigned the colour* a*. In other words, exactly one of the literals* $l_{\alpha\beta}$ *and* $l_{\gamma\delta}$ *in the formula* E *is assigned the truth value "false". On the other hand, the clauses of the form* $x_{\alpha 2} \neq x_{\alpha 3} \vee x_{\alpha 2} = a \vee x_{\alpha 3} = a$ *serve the following purpose: recall that the graph* G *contains a triangle with vertices* $\{y_{\alpha 1}, x_{\alpha 2}, x_{\alpha 3}\}$ *for each conjunction* $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ *where both* $l_{\alpha 2}$ *and* $l_{\alpha 3}$ *are literals over* P*. Then the clause* $x_{\alpha 2} \neq x_{\alpha 3} \vee x_{\alpha 2} = a \vee x_{\alpha 3} = a$ *guarantees that* G *has a valid* 3*-colouring, iff in every triangle at least one vertex is assigned the colour* a*. Actually the edge between the two vertices with labels* $x_{\alpha 2}$ *and*

$x_{\alpha 3}$ is omitted. Hence, the case $x_{\alpha 2} = x_{\alpha 3} = a$ is okay. However, the cases $x_{\alpha 2} = x_{\alpha 3} = b$ and $x_{\alpha 2} = x_{\alpha 3} = c$ would cause a problem, since then the triangle $\{y_{\alpha 1}, x_{\alpha 2}, x_{\alpha 3}\}$ would have a valid 3-colouring where the colour a is possibly not used.

Of course, this transformation can be done in polynomial time. It, therefore, only remains to prove the equivalence of the two problem instances, i.e. $\exists(p_1, \ldots, p_k)\, \forall(r_1, \ldots, r_l) E$ is satisfiable, iff $\exists\vec{x}\,\forall\vec{y}\forall\vec{z}\,[C_0 \wedge \mathcal{C}] \approx \top$. Actually, we shall prove the complementary equivalence, i.e. for every truth assignment $\mathcal{I}$ on P there exists an extension $\mathcal{J}$ to $P \cup R$, such that E evaluates to "false" in $\mathcal{J}$, iff for every ground substitution σ on $\vec{x}$, there exists a ground substitution τ on $\vec{y} \cup \vec{z}$, such that $\neg(C_0 \wedge \mathcal{C})\sigma\tau \approx \top$ holds.

Before we give a formal proof of this equivalence, we illustrate the main idea of this problem reduction by means of the following example:

Example 4.1. Consider the following Boolean formula with two alternating blocks of quantifiers:

$$\exists(p_1, p_2)\, \forall(r_1, r_2, r_3, r_4)[(\neg r_1 \wedge p_1 \wedge p_2) \vee (r_2 \wedge r_1 \wedge \neg p_1) \vee (r_2 \wedge \neg r_3 \wedge r_4)].$$

Then the clause C_0 and the conjunction $\mathcal{C}$ of clauses have the following form:

$$\begin{aligned} C_0 \;\equiv\;& y_{11} = x_{12} \vee y_{11} = x_{13} \vee \\ & y_{21} = y_{22} \vee y_{21} = x_{23} \vee y_{22} = x_{23} \vee \\ & y_{31} = y_{32} \vee y_{31} = y_{33} \vee y_{32} = y_{33} \vee \\ & z_1 = z_2 \vee z_1 = z_3 \vee z_2 = z_3 \vee \\ & z_1 = a \vee z_2 = y_{22} \vee z_3 = y_{11} \\ \mathcal{C} \;\equiv\;& (x_{12} = a \vee x_{23} = a) \wedge (x_{12} \neq a \vee x_{23} \neq a) \wedge \\ & (x_{12} \neq x_{13} \vee x_{12} = a \vee x_{13} = a). \end{aligned}$$

Recall that the clauses in $\mathcal{C}$ contain only variables from $\vec{x}$. Hence, for any ground substitution σ on $\vec{x}$, the conjunction $\mathcal{C}\sigma$ of clauses is either trivially true or trivially false. Then there is a one-to-one correspondence between truth assignments $\mathcal{I}$ on the propositional literals p_i and $\neg p_i$ in the Boolean formula E and ground substitutions σ on $\vec{x}$, for which $\mathcal{C}\sigma$ is trivially true, i.e. given a truth assignment $\mathcal{I}$, we can define the following ground substitution σ. Recall that $\vec{x}$ contains no variable of the form $x_{\alpha 1}$, since we assume throughout this paper that every conjunction in the Boolean formula E contains at least one literal over R and, in each conjunction, the literals over R stand in front of the literals over P:

$$\sigma(x_{\alpha 2}) = \begin{cases} a & \text{if } \mathcal{I}(l_{\alpha 2}) = \text{“false”} \\ b & \text{if } \mathcal{I}(l_{\alpha 2}) = \text{“true”} \end{cases} \qquad \sigma(x_{\alpha 3}) = \begin{cases} a & \text{if } \mathcal{I}(l_{\alpha 3}) = \text{“false”} \\ c & \text{if } \mathcal{I}(l_{\alpha 3}) = \text{“true”}. \end{cases}$$

Likewise, if σ is a ground substitution on $\vec{x}$, for which $\mathcal{C}\sigma$ is trivially true, then we can define the following truth assignment $\mathcal{I}$ on the propositional literals $l_{\alpha\beta}$ over P:

$$\mathcal{I}(l_{\alpha\beta}) = \begin{cases} \text{“true”} & \text{if there exists a literal } l_{\gamma\delta} \text{ in } E, \text{ such that} \\ & \quad l_{\alpha\beta} = l_{\gamma\delta} \text{ and } x_{\gamma\delta}\sigma \neq a \text{ hold} \\ \text{“false”} & \text{otherwise.} \end{cases}$$

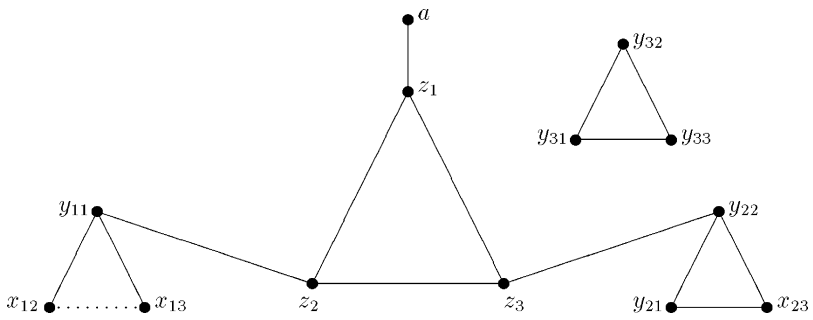

Fig. 2. Graph corresponding to the negated clause $\neg C_0$ from Example 4.1.

It can be easily checked that $\mathcal{I}$ is actually well-defined, i.e. if a literal occurs more than once in E, then it is always assigned the same truth value by $\mathcal{I}$. Moreover, complementary literals in E are indeed assigned complementary truth values.

Now suppose that we actually have a ground substitution σ on $\vec{x}$, such that all of the clauses in $\mathcal{C}\sigma$ are trivially true. As has already been mentioned above, the negated clause $\neg C_0$ corresponds to a graph 3*-colourability problem. In Fig.* 2*, this graph is displayed for the clause C_0 in the Example* 4.1 *above. The dotted line between x_{12} and x_{13} was inserted so as to visualize the one-to-one correspondence between the conjunctions in the Boolean formula E and the small triangles in the graph.*

Note that the negated clause $\neg C_0\sigma$ corresponds to a graph 3*-colourability problem, where the colour of some of the vertices has already been fixed. In order to prove the correctness of the problem reduction in Theorem* 4.2*, we basically have to show the following chain of equivalences: the ground substitution σ on $\vec{x}$ can be extended to $\sigma \cup \tau$ on $\vec{x} \cup \vec{y} \cup \vec{z}$, such that $\neg C_0(\sigma\tau) \approx \top$ holds. $\Leftrightarrow$ There exists a valid* 3*-colouring of the graph corresponding to $\neg C_0\sigma$, such that in every small triangle, at least one vertex is coloured by a. $\Leftrightarrow \mathcal{I}$ can be extended to an assignment $\mathcal{J}$ on $P \cup R$, such that in every conjunction of E at least one literal evaluates to "false". $\Leftrightarrow \mathcal{I}$ can be extended to an assignment $\mathcal{J}$ on $P \cup R$, such that E evaluates to "false" in $\mathcal{J}$.*

We are now ready to give a formal proof of the correctness of the problem reduction in Theorem 4.2, i.e. for every truth assignment $\mathcal{I}$ on P there exists an extension $\mathcal{J}$ to $P \cup R$, such that E evaluates to "false" in $\mathcal{J}$, iff for every ground substitution σ on $\vec{x}$, there exists a ground substitution τ on $\vec{y} \cup \vec{z}$, such that $\neg(C_0 \wedge \mathcal{C})\sigma\tau \approx \top$ holds:

Proof (Continuation). *"Only if"-direction*: Suppose that for every ground substitution σ on $\vec{x}$, there exists a ground substitution τ on $\vec{y} \cup \vec{z}$, such that $\neg(C_0 \wedge \mathcal{C})\sigma\tau \approx \top$ holds. Now let $\mathcal{I}$ be an arbitrary truth assignment on P. We have to show that then $\mathcal{I}$ can be extended to an assignment $\mathcal{J}$ on $P \cup R$, such that every conjunction $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ in E evaluates to "false" in $\mathcal{J}$. From $\mathcal{I}$ we define the following ground substitution σ on the first-order variables $x_{\alpha\beta} \in \vec{x}$. Recall that $\vec{x}$ contains no variable of the form $x_{\alpha 1}$, since we assume throughout this paper that every conjunction in the Boolean formula E contains at least

one literal over R and, in each conjunction, the literals over R stand in front of the literals over P:

$$\sigma(x_{\alpha 2}) = \begin{cases} a & \text{if } \mathcal{I}(l_{\alpha 2}) = \text{“false”} \\ b & \text{if } \mathcal{I}(l_{\alpha 2}) = \text{“true”} \end{cases} \qquad \sigma(x_{\alpha 3}) = \begin{cases} a & \text{if } \mathcal{I}(l_{\alpha 3}) = \text{“false”} \\ c & \text{if } \mathcal{I}(l_{\alpha 3}) = \text{“true”}. \end{cases}$$

By assumption, there exists a ground substitution τ on $\vec{y} \cup \vec{z}$, such that $\neg(C_0 \wedge \mathcal{C})\sigma\tau \approx \neg C_0\sigma\tau \vee \neg\mathcal{C}\sigma\tau \approx \top$ holds. Actually, $\neg\mathcal{C}\sigma\tau \not\approx \top$ holds, since $C_i\sigma\tau \approx C_i\sigma \approx \top$ holds for every clause $C_i \in \mathcal{C}$ by the above definition of σ. This can be seen as follows:

- If $(l_{\alpha\beta}, l_{\gamma\delta})$ is a pair of complementary literals over P, then one of these literals evaluates to “false” in $\mathcal{I}$ and one evaluates to “true” in $\mathcal{I}$. Thus, by the definition of σ, all clauses of the form $(x_{\alpha\beta} = a \vee x_{\gamma\delta} = a)\sigma$ and $(x_{\alpha\beta} \neq a \vee x_{\gamma\delta} \neq a)\sigma$ in $\mathcal{C}\sigma$ are trivially true.
- Likewise, all clauses of the form $(x_{\alpha 2} \neq x_{\alpha 3} \vee x_{\alpha 2} = a \vee x_{\alpha 3} = a)\sigma$ in $\mathcal{C}\sigma$ are trivially true by the definition of σ.

Hence, there must exist a ground substitution τ on $\vec{y} \cup \vec{z}$, such that $\neg C_0\sigma\tau \approx \top$ holds. From τ we construct the following extension $\mathcal{J}$ of $\mathcal{I}$ to $P \cup R$:

$$\mathcal{J}(l_{\alpha\beta}) = \begin{cases} \text{“false”} & \text{if there exists a literal } l_{\gamma\delta} \text{ in} E, \text{ such that} \\ & \quad l_{\alpha\beta} = l_{\gamma\delta} \text{ and } y_{\gamma\delta}\tau = a \\ \text{“undefined”} & \text{otherwise.} \end{cases}$$

We claim that, on the one hand, $\mathcal{J}$ is well-defined and, on the other hand, every conjunction in E evaluates to “false” in $\mathcal{J}$:

In order to show that $\mathcal{J}$ is well-defined, we only have to prove that no two complementary literals are assigned the truth value “false” by $\mathcal{J}$. Suppose on the contrary that there exists a pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R with $\mathcal{J}(l_{\alpha\beta}) = \mathcal{J}(l_{\gamma\delta}) =$ “false”. Then, by the definition of $\mathcal{J}$, there exists a literal $l_{\alpha'\beta'}$ which is identical to $l_{\alpha\beta}$, such that $y_{\alpha'\beta'}\tau = a$ holds. Likewise, there exists a literal $l_{\gamma'\delta'}$ which is identical to $l_{\gamma\delta}$ with $y_{\gamma'\delta'}\tau = a$. Note that then also $l_{\alpha'\beta'}$ and $l_{\gamma'\delta'}$ are complementary literals. Hence, for the pair $(l_{\alpha'\beta'}, l_{\gamma'\delta'})$ of complementary literals, $\neg C_0\sigma\tau$ contains the six disequations $z_1\tau \neq z_2\tau$, $z_1\tau \neq z_3\tau$, $z_2\tau \neq z_3\tau$, $z_1\tau \neq a$, $z_2\tau \neq y_{\alpha'\beta'}\tau$ and $z_3\tau \neq y_{\gamma'\delta'}\tau$ which are all trivially true by assumption. It is easy to check that then either $y_{\alpha'\beta'}\tau \neq a$ or $y_{\gamma'\delta'}\tau \neq a$ must hold, which contradicts the above considerations.

In order to show that every conjunction $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ in E evaluates to “false” in $\mathcal{J}$ we have to show for every $\alpha \in \{1, \ldots, n\}$ that at least one literal $l_{\alpha\beta}$ evaluates to “false” in $\mathcal{J}$. We distinguish the following cases (which correspond to the cases in the definition of the clause C_0):

- *Case 1*: If all of the literals $l_{\alpha 1}$, $l_{\alpha 2}$ and $l_{\alpha 3}$ are literals over R (i.e. they are of the form r_i or $\neg r_i$), then $\neg C_0\sigma\tau$ contains the three disequations $y_{\alpha 1}\tau \neq y_{\alpha 2}\tau$, $y_{\alpha 1}\tau \neq y_{\alpha 3}\tau$ and $y_{\alpha 2}\tau \neq y_{\alpha 3}\tau$. By assumption, all of these disequations are trivially true. Hence, at least one of the variables $y_{\alpha 1}$, $y_{\alpha 2}$ and $y_{\alpha 3}$ is instantiated to a by τ. But then, by the definition of $\mathcal{J}$, at least one of the literals $l_{\alpha 1}$, $l_{\alpha 2}$ and $l_{\alpha 3}$ evaluates to “false” in $\mathcal{J}$.
- *Case 2*: If $l_{\alpha 1}$ and $l_{\alpha 2}$ are literals over R and $l_{\alpha 3}$ is a literal over P, then $\neg C_0\sigma\tau$ contains the three disequations $y_{\alpha 1}\tau \neq y_{\alpha 2}\tau$, $y_{\alpha 1}\tau \neq x_{\alpha 3}\sigma$ and $y_{\alpha 2}\tau \neq x_{\alpha 3}\sigma$. If

$x_{\alpha 3}$ is instantiated to a by σ then, by the definition of σ, the literal $l_{\alpha 3}$ evaluates to "false" in $\mathcal{I}$ and, therefore, also in $\mathcal{J}$. On the other hand, if $x_{\alpha 3}\sigma$ is different from a, then either $y_{\alpha 1}\tau$ or $y_{\alpha 2}\tau$ must be equal to a. But then, by the definition of $\mathcal{J}$, either $l_{\alpha 2}$ or $l_{\alpha 3}$ evaluates to "false" in $\mathcal{J}$.

- *Case 3*: If $l_{\alpha 1}$ is a literal over R and $l_{\alpha 2}$ and $l_{\alpha 3}$ are literals over P, then $\neg C_0\sigma\tau$ contains the two disequations $y_{\alpha 1}\tau \not\approx x_{\alpha 2}\sigma$ and $y_{\alpha 1}\tau \not\approx x_{\alpha 3}\sigma$, which are both trivially true by assumption. If $x_{\alpha 2}\sigma$ or $x_{\alpha 3}\sigma$ is equal to a, then the corresponding literal $l_{\alpha 2}$ or $l_{\alpha 3}$ evaluates to "false" in $\mathcal{I}$ and, therefore, also in $\mathcal{J}$. On the other hand, if both $x_{\alpha 2}\sigma$ and $x_{\alpha 3}\sigma$ are different from a then, by the definition of σ, the equalities $x_{\alpha 2}\sigma = b$ and $x_{\alpha 3}\sigma = c$ hold. But then $y_{\alpha 1}\tau = a$ must hold in order to validate the above two disequations. Hence, $l_{\alpha 1}$ evaluates to "false" in $\mathcal{J}$.

"if"-direction: Suppose that for every truth assignment $\mathcal{I}$ on P there exists an extension $\mathcal{J}$ to $P \cup R$, such that E evaluates to "false" in $\mathcal{J}$. Moreover, let σ be an arbitrary ground substitution on $\vec{x}$. We have to show that there exists a ground substitution τ on $\vec{y} \cup \vec{z}$, such that $\neg(C_0 \wedge \mathcal{C})\sigma\tau \approx \top$ holds. Note that $\mathcal{C}$ contains only variables from $\vec{x}$. Hence, if $\neg\mathcal{C}\sigma \approx \top$ holds, then $\neg\mathcal{C}\sigma\tau \approx \top$ and $\neg(C_0 \wedge \mathcal{C})\sigma\tau \approx \top$ clearly also hold for any substitution τ on $\vec{y} \cup \vec{z}$ and we are done. It therefore only remains to consider the case where $\mathcal{C}\sigma \approx \top$ holds. Then we define the following truth assignment $\mathcal{I}$ on the propositional literals in P:

$$\mathcal{I}(l_{\alpha\beta}) = \begin{cases} \text{"true"} & \text{if there exists a literal } l_{\gamma\delta} \text{ in } E, \text{ such that} \\ & \quad l_{\alpha\beta} = l_{\gamma\delta} \text{ and } x_{\gamma\delta}\sigma \not\approx a \text{ hold} \\ \text{"false"} & \text{otherwise.} \end{cases}$$

First of all note that this truth assignment is well defined, i.e.

- If $(l_{\alpha\beta}, l_{\gamma\delta})$ is a pair of identical literals over P then, by the definition of $\mathcal{I}$, we clearly have $\mathcal{I}(l_{\alpha\beta}) = \mathcal{I}(l_{\gamma\delta})$.
- Now let $(l_{\alpha\beta}, l_{\gamma\delta})$ be a pair of complementary literals over P. We have to show that they are assigned distinct truth values in $\mathcal{I}$. Suppose on the contrary that either both are assigned the value "true" or both are assigned the value "false". Actually, if both literals $l_{\alpha\beta}$ and $l_{\gamma\delta}$ are assigned the value "false" then, by the definition of $\mathcal{I}$, both $x_{\alpha\beta}\sigma = a$ and $x_{\gamma\delta}\sigma = a$ hold. However, this is impossible, since we consider the case where $\mathcal{C}\sigma \approx \top$ holds and $\mathcal{C}\sigma$ contains the clause $x_{\alpha\beta}\sigma \not\approx a \vee x_{\gamma\delta}\sigma \not\approx a$. So suppose that both literals $l_{\alpha\beta}$ and $l_{\gamma\delta}$ are assigned the value "true". Then, by the definition of $\mathcal{I}$, there exists a literal $l_{\alpha'\beta'}$ which is identical to $l_{\alpha\beta}$, such that $x_{\alpha'\beta'}\sigma \not\approx a$ holds. Likewise, there exists a literal $l_{\gamma'\delta'}$ which is identical to $l_{\gamma\delta}$ with $x_{\gamma'\delta'}\sigma \not\approx a$. Of course, $(l_{\alpha'\beta'}, l_{\gamma'\delta'})$ is also a pair of complementary literals. But then $\mathcal{C}\sigma$ contains the clause $x_{\alpha'\beta'}\sigma = a \vee x_{\gamma'\delta'}\sigma = a$, which is trivially true by assumption. So we have again a contradiction.

By assumption, there exists an extension $\mathcal{J}$ of $\mathcal{I}$ to the propositional variables in $P \cup R$, such that E evaluates to "false" in $\mathcal{J}$. From $\mathcal{J}$ we construct a ground substitution τ on $\vec{y} \cup \vec{z}$ for which $\neg C_0\sigma\tau \approx \top$ holds. For this construction we again distinguish the cases from the definition of the clause C_0.

- *Case 1*: If all of the literals $l_{\alpha 1}, l_{\alpha 2}$ and $l_{\alpha 3}$ are literals over R, then $\neg C_0\sigma$ contains the three disequations $y_{\alpha 1} \neq y_{\alpha 2}$, $y_{\alpha 1} \neq y_{\alpha 3}$ and $y_{\alpha 2} \neq y_{\alpha 3}$. By assumption, at least one of the literals $l_{\alpha 1}, l_{\alpha 2}$ and $l_{\alpha 3}$ evaluates to "false" in $\mathcal{J}$, $l_{\alpha\beta}$ say. Then we set $y_{\alpha\beta}\tau = a$ and require that τ has to instantiate the remaining two first-order variables in $\{y_{\alpha 1}, y_{\alpha 2}, y_{\alpha 3}\}$ to b and c, respectively. Note that then all of the disequations $y_{\alpha 1}\tau \neq y_{\alpha 2}\tau$, $y_{\alpha 1}\tau \neq y_{\alpha 3}\tau$ and $y_{\alpha 2}\tau \neq y_{\alpha 3}\tau$ in $\neg C_0\sigma\tau$ are trivially true.
- *Case 2*: If $l_{\alpha 1}$ and $l_{\alpha 2}$ are literals over R and $l_{\alpha 3}$ is a literal over P, then $\neg C_0\sigma$ contains the three disequations $y_{\alpha 1} \neq y_{\alpha 2}$, $y_{\alpha 1} \neq x_{\alpha 3}\sigma$ and $y_{\alpha 2} \neq x_{\alpha 3}\sigma$. If $x_{\alpha 3}\sigma = a$ holds, then we can define τ on $\{y_{\alpha 1}, y_{\alpha 2}\}$ in such a way that one of the variables is instantiated to b and the other one to c. On the other hand, if $x_{\alpha 3}\sigma \neq a$ holds then, by the definition of $\mathcal{I}$, the literal $l_{\alpha 3}$ evaluates to "true" in $\mathcal{I}$ and, therefore, also in $\mathcal{J}$. However, we know that at least one of the literals $l_{\alpha 1}, l_{\alpha 2}$ and $l_{\alpha 3}$ evaluates to "false" in $\mathcal{J}$. Hence, at least one of the literals $l_{\alpha 1}$ and $l_{\alpha 2}$ evaluates to "false" in $\mathcal{J}$, $l_{\alpha\beta}$ say. Then we define τ on $\{y_{\alpha 1}, y_{\alpha 2}\}$ in such a way that $y_{\alpha\beta}$ is instantiated to a and the other variable is either instantiated to b (if $x_{\alpha 3}\sigma = c$ holds) or to c (in the case of $x_{\alpha 3}\sigma = b$), respectively. Thus, in any case, the resulting disequations $y_{\alpha 1}\tau \neq y_{\alpha 2}\tau$, $y_{\alpha 1}\tau \neq x_{\alpha 3}\sigma$ and $y_{\alpha 2}\tau \neq x_{\alpha 3}\sigma$ in $\neg C_0\sigma\tau$ are again true.
- *Case 3*: If $l_{\alpha 1}$ is a literal over R and both $l_{\alpha 2}$ and $l_{\alpha 3}$ are literals over P, then $\neg C_0\sigma$ contains the two disequations $y_{\alpha 1} \neq x_{\alpha 2}\sigma$ and $y_{\alpha 1} \neq x_{\alpha 3}\sigma$. Again we can define τ in such a way that both disequations $y_{\alpha 1}\tau \neq x_{\alpha 2}\sigma$ and $y_{\alpha 1}\tau \neq x_{\alpha 3}\sigma$ are trivially true and, furthermore, $y_{\alpha 1}\tau$ is equal to a, only if $l_{\alpha 1}$ evaluates to "false" in $\mathcal{J}$.

It only remains to provide an appropriate definition of τ on $\vec{z}$: Let $(l_{\alpha\beta}, l_{\gamma\delta})$ be a pair of complementary literals over R. Then either $l_{\alpha\beta}$ or $l_{\gamma\delta}$ is not assigned the value "false" by $\mathcal{J}$. Hence, by the above definition of τ on $\vec{y}$, either $y_{\alpha\beta}\tau$ or $y_{\gamma\delta}\tau$ is different from a. By distinguishing all possible cases of $(y_{\alpha\beta}\tau, y_{\gamma\delta}\tau) \in \{a, b, c\}^2 - \{(a, a)\}$ we have to check that the values of $z_1\tau$, $z_2\tau$ and $z_3\tau$ can be chosen in such a way that all of the resulting six disequations $z_1\tau \neq z_2\tau$, $z_1\tau \neq z_3\tau$, $z_2\tau \neq z_3\tau$, $z_1\tau \neq a$, $z_2\tau \neq y_{\alpha\beta}\tau$ and $z_3\tau \neq y_{\gamma\delta}\tau$ in $\neg C_0\sigma\tau$ are trivially true, i.e.

Case 1: For $y_{\alpha\beta}\tau = a$ and $y_{\gamma\delta}\tau = b$, we set $z_1\tau = b$, $z_2\tau = c$ and $z_3\tau = a$.

Case 2: For $y_{\alpha\beta}\tau = a$ and $y_{\gamma\delta}\tau = c$, we set $z_1\tau = c$, $z_2\tau = b$ and $z_3\tau = a$.

Case 3: For $y_{\alpha\beta}\tau = b$ and $y_{\gamma\delta}\tau = a$, we set $z_1\tau = b$, $z_2\tau = a$ and $z_3\tau = c$.

$\vdots$

Case 8: For $y_{\alpha\beta}\tau = c$ and $y_{\gamma\delta}\tau = c$, we set $z_1\tau = c$, $z_2\tau = a$ and $z_3\tau = b$.

Note that by considering all pairs $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R that occur in E, we finally end up with a ground substitution τ on $\vec{y} \cup \vec{z}$ for which every conjunct in $\neg C_0\sigma\tau$ is trivially true. □

In order to extend the Σ_2^p-hardness proof idea from a domain with exactly three elements to an arbitrary finite domain with at least three elements, we first extend the 3-QSAT$_2$ problem from Section 2.4 to the following K-QSAT$_2$ problem:

K-QSAT$_2$

Input: A triple (P, R, E), such that $P = \{p_1, \ldots, p_k\}$ and $R = \{r_1, \ldots, r_l\}$ are disjoint sets of propositional variables and $E = (l_{11} \wedge \cdots \wedge l_{1K}) \vee \cdots \vee (l_{n1} \wedge \cdots \wedge l_{nK})$ is a Boolean formula with propositional variables in $P \cup R$.

Question: Is the quantified Boolean sentence $\exists(p_1, \ldots, p_k)\, \forall(r_1, \ldots, r_l) E$ satisfiable?

Of course, the K-QSAT$_2$ problem is Σ_2^p-hard for every $K \geq 3$. Moreover, we may again restrict the form of E like in Section 2.4, i.e. no conjunction $C_i = l_{i1} \wedge \cdots \wedge l_{iK}$ contains a pair of complementary literals. Every conjunction C_i contains at least one literal over R. Finally, in each conjunction C_i, the literals over R stand in front of the literals over P.

We thus have all the ingredients for proving the following result:

Theorem 4.3 (CNF Over a K-Element Domain with $K \geq 3$). *Let D be a finite domain with at least three elements. Then the satisfiability problem for equational problems in CNF over D is Σ_2^p-complete.*

Proof (Sketch). Again, the *Σ_2^p-membership* is clear by Theorem 4.1. In order to prove also the *Σ_2^p-hardness*, the problem reduction in Theorem 4.2 has to be extended to an arbitrary finite domain D with $K \geq 3$ elements in the following way: let $\mathcal{Q} = (P, R, E)$ be an instance of the K-QSAT$_2$ problem, where $P = \{p_1, \ldots, p_k\}$ and $R = \{r_1, \ldots, r_l\}$ are sets of propositional variables and $E = (l_{11} \wedge \cdots \wedge l_{1K}) \vee \cdots \vee (l_{n1} \wedge \cdots \wedge l_{nK})$ is a Boolean formula with propositional variables in $P \cup R$. Then we define the equational problem $\mathcal{P} \equiv \exists\vec{x}\,\forall\vec{y}\,\forall\vec{z}\,[C_0 \wedge \mathcal{C}]$ over $D = \{a_1, \ldots, a_K\}$ as follows:

Variables:

- "$\vec{x}$": For every literal $l_{\alpha\beta}$ over P, there is a first-order variable $x_{\alpha\beta}$ in $\vec{x}$.
- "$\vec{y}$": For every literal $l_{\alpha\beta}$ over R, there is a first-order variable $y_{\alpha\beta}$ in $\vec{y}$.
- "$\vec{z}$": For every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R with $l_{\alpha\beta} \in R$ and $l_{\gamma\delta} = \neg l_{\alpha\beta}$, there are K first-order variables $z_1, \ldots, z_K$ in $\vec{z}$.

All of the variables in $\vec{x}$, $\vec{y}$ and $\vec{z}$ are assumed to be pairwise distinct.

Clause C_0: Analogously to the special case $K = 3$, we define the "big clause" C_0 in such a way that it contains some information about the conjunctions of E and the complementary literals over R occurring in E, i.e. let $l_{\alpha 1} \wedge \cdots \wedge l_{\alpha j} \wedge \cdots \wedge l_{\alpha K}$ be the αth conjunction of E with $1 \leq j \leq K$, such that $l_{\alpha 1}, \ldots, l_{\alpha j}$ are literals over R and $l_{\alpha(j+1)}, \ldots, l_{\alpha K}$ are literals over P. Then C_0 contains the following disjuncts:

- For every β and γ with $1 \leq \beta < \gamma \leq j$, there is an equation $y_{\alpha\beta} = y_{\alpha\gamma}$ in C_0.
- For every β and γ with $1 \leq \beta \leq j < \gamma \leq K$, there is an equation $y_{\alpha\beta} = x_{\alpha\gamma}$ in C_0.

Moreover, for every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R, where $l_{\alpha\beta} \in R$ and $l_{\gamma\delta} = \neg l_{\alpha\beta}$ hold, C_0 contains the following disjuncts:

- For every i and j with $1 \leq i < j \leq K$, there is an equation $z_i = z_j$ in C_0.
- For every i with $1 \leq i \leq K - 2$, there is an equation $z_i = a_1$ in C_0.
- Finally, C_0 contains the equations $z_{K-1} = y_{\alpha\beta}$ and $z_K = y_{\gamma\delta}$.

No further disjuncts are contained in C_0.

The above definition is based on the same idea as the construction of the clause C_0 in the proof of Theorem 4.2. Again the negated clause $\neg C_0$ corresponds to a colouring problem of a graph G. Of course, this time we have to deal with a K-element domain D and, hence, we are interested in a K-colouring of G rather than a 3-colouring. Recall that in Theorem 4.2, every conjunction $l_{\alpha 1} \wedge l_{\alpha 2} \wedge l_{\alpha 3}$ in the Boolean formula E was encoded by a triangle in the graph G, where the edges between any two vertices with labels from $\vec{x}$ were omitted. More generally, we now have to deal with conjunctions $l_{\alpha 1} \wedge \cdots \wedge l_{\alpha K}$ which are encoded by K-cliques in the graph G, where again the edges between any two vertices with labels from $\vec{x}$ are omitted. The truth value "false" of a literal $l_{\alpha\beta}$ in E is now encoded by the element $a_1 \in D$. Again, for every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over R, we have to make sure that not both literals $l_{\alpha\beta}$ and $l_{\gamma\delta}$ are assigned the truth value "false". Analogously to the construction in Theorem 4.2, the clause C_0 contains equations involving the variables $z_1, \ldots, z_K$ for this purpose. In analogy to the gadget in Fig. 1, the subgraph corresponding to the disequations in $\neg C_0$ involving the variables $z_1, \ldots, z_K$ consists of a K-clique with vertices $z_1, \ldots, z_K$, such that $K - 2$ of these vertices are adjacent to an additional vertex with colour a_1 and the remaining two vertices are connected to the vertices with labels $y_{\alpha\beta}$ and $y_{\gamma\delta}$, respectively. Hence, this subgraph has a valid K-colouring, iff at least one of the vertices with labels $y_{\alpha\beta}$ and $y_{\gamma\delta}$ is assigned a colour different from a_1. The latter condition is equivalent to the requirement that not both literals $l_{\alpha\beta}$ and $l_{\gamma\delta}$ in E are assigned the truth value "false".

Clauses in $\mathcal{C}$: The conjunction $\mathcal{C}$ of clauses with variables from $\vec{x}$ consists of the following clauses:

- For every pair $(l_{\alpha\beta}, l_{\gamma\delta})$ of complementary literals over P, where $l_{\alpha\beta} \in P$ and $l_{\gamma\delta} = \neg l_{\alpha\beta}$ hold, $\mathcal{C}$ contains the following two clauses: $x_{\alpha\beta} = a_1 \vee x_{\gamma\delta} = a_1$ and $x_{\alpha\beta} \neq a_1 \vee x_{\gamma\delta} \neq a_1$.
- Suppose that the αth conjunction in E is of the form $l_{\alpha 1} \wedge \cdots \wedge l_{\alpha j} \wedge \cdots \wedge l_{\alpha K}$ with $1 \leq j \leq K - 2$, such that $l_{\alpha 1}, \ldots, l_{\alpha j}$ are literals over R and $l_{\alpha(j+1)}, \ldots, l_{\alpha K}$ are literals over P (i.e. the αth conjunction in E contains at least two literals over P). Then, for every pair (β, γ) with $j < \beta < \gamma \leq K$, the set $\mathcal{C}$ contains the clause $x_{\alpha\beta} \neq x_{\alpha\gamma} \vee x_{\alpha\beta} = a_1 \vee x_{\alpha\gamma} = a_1$.

No further clauses are in $\mathcal{C}$.

The two kinds of clauses in $\mathcal{C}$ serve the same purpose as the corresponding clauses in the proof of Theorem 4.2, namely: The clauses of the form $x_{\alpha\beta} = a_1 \vee x_{\gamma\delta} = a_1$ and $x_{\alpha\beta} \neq a_1 \vee x_{\gamma\delta} \neq a_1$ make sure that exactly one of the literals $l_{\alpha\beta}$ and $l_{\gamma\delta}$ in E is assigned the truth value "false". Now recall that for every conjunction $l_{\alpha 1} \wedge \cdots \wedge l_{\alpha K}$ in E, the graph G corresponding to the negated clause C_0 contains a K-clique with vertices in $\vec{x} \cup \vec{y}$, where the edges between any two vertices from $\vec{x}$ are omitted. Then the clauses of the form $x_{\alpha\beta} \neq x_{\alpha\gamma} \vee x_{\alpha\beta} = a_1 \vee x_{\alpha\gamma} = a_1$ in $\mathcal{C}$ guarantee that G has a valid K-colouring, iff in every K-clique at least one vertex is assigned the colour a_1.

Again this problem reduction can be done in polynomial time. Hence, it only remains to prove the equivalence of the two problem instances, i.e. $\exists(p_1, \ldots, p_k)\, \forall(r_1, \ldots, r_l) E$ is satisfiable, iff $\exists\vec{x} \forall\vec{y} \forall\vec{z} [C_0 \wedge \mathcal{C}] \approx \top$ holds. The extension of the equivalence proof

from Theorem 4.2 to the case of an arbitrary finite domain with $K \geq 3$ elements is straightforward and is therefore omitted here. □

In order to arrive at a complete complexity analysis of equational problems over a finite domain, it remains to investigate equational problems in CNF over a domain with only two elements. Actually, in this case the problem reduction in Theorems 4.2 and 4.3 does not work. Recall that the graph K-colourability problem is NP-hard only for $K \geq 3$. On the other hand, for $K = 2$, it is in P. Analogously, it can be shown for equational problems in CNF over a domain with K elements, that the satisfiability problem for $K = 2$ is one level lower in the polynomial hierarchy than for $K \geq 3$, i.e.

Theorem 4.4 (CNF Over a Two-Element Domain). *Let D be a domain with two elements. Then the satisfiability problem for equational problems in CNF over D is NP-complete.*

Proof. Recall from Theorem 3.1 that the *NP-hardness* holds even for equational problems in CNF with $\exists^*$-prefix. The *NP-membership* is due to the fact that (in contrast to a domain with $K \geq 3$ elements) the satisfiability of a purely existentially quantified conjunction of equations and disequations can be tested in polynomial time. Then the satisfiability of the equational problem $\mathcal{P} \equiv \exists\vec{x}\forall\vec{y}\mathcal{P}'$ can be tested by guessing a ground substitution σ on the variables $\vec{x}$ and checking in polynomial time that the resulting problem $\forall\vec{y}\mathcal{P}'\sigma$ is satisfiable (or, equivalently, that $\exists\vec{y}\neg\mathcal{P}'\sigma$ is unsatisfiable).

So it only remains to prove that the satisfiability of a purely existentially quantified conjunction of equations and disequations can indeed be tested in polynomial time. To this end, let $\mathcal{R} \equiv \exists\vec{x}\mathcal{Q}$ with $\mathcal{Q} \equiv s_1 = t_1 \wedge \cdots \wedge s_k = t_k \wedge u_1 \neq v_1 \wedge \cdots \wedge u_l \neq v_l$ be an equational problem over the domain $D = \{a, b\}$. Then we may use the following algorithm to check whether there exists a substitution σ on $\vec{x}$, such that $\mathcal{Q}\sigma \approx \top$ holds:

Initialization: We initialize σ to the empty substitution, i.e. $\sigma := \{\}$.

Rule 1: If $\mathcal{Q}\sigma$ contains a conjunct of the form $x_i = a$, $a = x_i$, $x_i \neq b$ or $b \neq x_i$ for some variable $x_i \in \vec{x}$, then we extend σ to $\sigma \cup \{x_i \leftarrow a\}$.

Rule 2: If $\mathcal{Q}\sigma$ contains a conjunct of the form $x_i = b$, $b = x_i$, $x_i \neq a$ or $a \neq x_i$, then we set $\sigma := \sigma \cup \{x_i \leftarrow b\}$.

Rule 3: If none of the above two rules can be applied and if there still occurs a variable $x_j \in \vec{x}$ in $\mathcal{Q}\sigma$, then we may set $\sigma := \sigma \cup \{x_j \leftarrow a\}$.

End Condition: This algorithm terminates when $\mathcal{Q}\sigma$ is a conjunction of ground equations and disequations. If all these conjuncts are trivially true (i.e. we only have conjuncts of the form $a = a$, $b = b$, $a \neq b$ and $b \neq a$), then $\mathcal{R}$ is satisfiable. Otherwise it is not.

This algorithm clearly works in deterministically polynomial time. The correctness can be easily seen by the analogy with a straightforward decision algorithm for the graph 2-colourability problem, i.e. if there exists a vertex V in a connected component of the given graph, such that V has already been assigned some colour, then we assign the opposite colour to all adjacent vertices of V. This step corresponds to Rules 1 and 2 above. On the other hand, if there exists a connected component of the given graph in which no vertex has

been assigned a colour yet, then we may select any vertex from this connected component and assign any colour to it. This step corresponds to Rule 3. □

5. Equational problems over an infinite domain

The Σ_2^p-hardness proof in Theorem 4.1 for equational problems in arbitrary form or in DNF can be literally taken over from a finite domain to an infinite domain. On the other hand, the Σ_2^p-membership in the case of DNF or arbitrary form over an infinite domain is by no means clear and has to be left as an open problem. The remainder of this section is therefore devoted to a complexity analysis of equational problems in CNF over an infinite domain. In Section 5.1, we provide a polynomial time transformation of equational problems in CNF into a much simpler syntactical form and in Section 5.2 we shall make use of this simple form to prove the NP-membership.

5.1. Parameter-free equations

The target of this section is a transformation of equational problems in CNF into the following form (which is similar to the normal form presented in Maher, 1988):

Definition (PFE-Form). An equational problem $\mathcal{P}$ is said to be in PFE-form (= *parameter-free equations*), iff it has the form

$$(\exists\vec{x})[(e_{11} \vee \cdots \vee e_{1k_1}) \vee (\forall\vec{y}_1)(\vec{z}_1 \neq \vec{t}_1)] \wedge \cdots \wedge [(e_{n1} \vee \cdots \vee e_{nk_n}) \vee (\forall\vec{y}_n)(\vec{z}_n \neq \vec{t}_n)],$$

such that the e_{ij}'s are equations, every $\vec{t}_i$ is a term tuple with variables in $\vec{y}_i$ and every $\vec{z}_i$ denotes a tuple of existentially quantified variables from $\vec{x}$.

Before we can start with the elimination of the parameters from the equations, we first have to simplify the disequations via the following lemma:

Lemma 5.1 (Simplification of Disequations). *Let D be an arbitrary domain and let $\mathcal{P} \equiv (\forall\vec{y})[\mathcal{R} \vee \vec{s} \neq \vec{t}]$ be an equational problem over D with free variables in $\vec{x}$, where $\mathcal{R}$ denotes an arbitrary equational problem. Then $\mathcal{P}$ can be simplified via unification in the following way:*

- *Case 1: If the equation $\vec{s} = \vec{t}$ is not unifiable, then $\mathcal{P} \approx \top$.*
- *Case 2: Suppose that $\vartheta = \{x_{i_1} \leftarrow u_1, \ldots, x_{i_k} \leftarrow u_k, y_{j_1} \leftarrow v_1, \ldots, y_{j_l} \leftarrow v_l\}$ is the mgu of $\vec{s} = \vec{t}$ and let the substitution η be defined as $\eta = \{y_{j_1} \leftarrow v_1, \ldots, y_{j_l} \leftarrow v_l\}$. Then $\mathcal{P}$ is equivalent to $\mathcal{P}' \equiv (\forall\vec{y})[\mathcal{R}\eta \vee x_{i_1} \neq u_1 \vee \cdots \vee x_{i_k} \neq u_k]$.*

Proof. In Case 1, there exists no substitution σ on the variables $\vec{x} \cup \vec{y}$, such that $\vec{s}\sigma = \vec{t}\sigma$. Hence, $\vec{s} \neq \vec{t} \approx \top$ and also $\mathcal{P} \approx \top$ clearly hold.

In Case 2, the equivalence $\vec{s} \neq \vec{t} \approx x_{i_1} \neq u_1 \vee \cdots \vee x_{i_k} \neq u_k \vee y_{j_1} \neq v_1 \vee \cdots \vee y_{j_l} \neq v_l$ holds by the definition of the mgu. Note that the variables y_{j_α} occur exactly once in the disequations thus produced. Hence, l applications of the U_2-rule recalled in Section 2.2 to the disequations $y_{j_1} \neq v_1, \ldots, y_{j_l} \neq v_l$ can be contracted to a single transformation step via the substitution η, i.e. $\mathcal{P} \approx (\forall\vec{y})[\mathcal{R}\eta \vee (x_{i_1} \neq u_1 \vee \cdots \vee x_{i_k} \neq u_k)\eta]$. Finally note that

the substitution η only needs to be applied to $\mathcal{R}$, since the variables y_{j_α} from the domain of η do not occur in the disequations $x_{i_\beta} \neq u_\beta$. □

The U_4-rule recalled in Section 2.2 allows the elimination of those parameters from the equations which do not occur in the disequations. However, the U_4-rule is very restrictive in that it requires the equations to be of the form $z = u$, where z is a variable. In the following lemma we show how all parameters not occurring in the disequations can be eliminated from the equations.

Lemma 5.2 (Elimination of Parameters not Occurring in the Disequations). *Let D be an infinite domain and let $\mathcal{P} \equiv (\forall \vec{y})(s_1 = t_1 \vee \cdots \vee s_m = t_m \vee \mathcal{R})$ be an equational problem over D with free variables in $\vec{x}$, such that every equation $s_i = t_i$ contains at least one parameter from $\vec{y}$ and $\mathcal{R}$ contains no parameter from $\vec{y} = \{y_1, \ldots, y_k\}$. Then the parameters $\vec{y}$ can be eliminated from $\mathcal{P}$ by the following rules:*

1. *Non-unifiable equations: If an equation $s_i = t_i$ is not unifiable, then $s_i = t_i$ may be deleted from $\mathcal{P}$.*
2. *Parameters not occurring in the mgu: If $s_i = t_i$ is unifiable with mgu ϑ and no parameter y_α occurs in ϑ (i.e. $\mathrm{dom}(\vartheta) \cap \vec{y} = \emptyset$ and $\mathrm{Var}(\mathrm{rg}(\vartheta)) \cap \vec{y} = \emptyset$), then every occurrence of every parameter $y_\alpha \in \vec{y}$ in $s_i = t_i$ may be replaced by an arbitrary constant symbol $a \in D$.*
3. *Parameters occurring in the mgu: If $s_i = t_i$ is unifiable with mgu ϑ and at least one parameter y_α occurs in ϑ (i.e. $\mathrm{dom}(\vartheta) \cap \vec{y} \neq \emptyset$ or $\mathrm{Var}(\mathrm{rg}(\vartheta)) \cap \vec{y} \neq \emptyset$), then the equation $s_i = t_i$ may be deleted from $\mathcal{P}$.*

Proof. Case 1 is obvious. Case 2 is also clear, since the mgu ϑ completely describes all solutions of an equation and replacing variables from outside the mgu does not change the mgu. Hence, we are only left with Case 3: Let I denote the indices of the equations to which Case 3 applies. Note that by Cases 1 and 2, all parameters can be eliminated from the equations $s_j = t_j$ for $j \notin I$. Hence, $\mathcal{P}$ is equivalent to $\mathcal{P}' \equiv (\forall \vec{y}) \bigvee_{i \in I} (s_i = t_i) \vee \mathcal{R}'$ for appropriately chosen $\mathcal{R}'$ such that $\mathcal{R}'$ contains no parameter from $\vec{y}$. For every $i \in I$ let $\vartheta_i = \{z_i \leftarrow u_i\} \cup \eta_i$ denote the mgu of $s_i = t_i$ such that some parameter $y_{\alpha_i} \in \vec{y}$ occurs in z_i or u_i. Then, for $\mathcal{P}'' \equiv (\forall \vec{y}) \bigvee_{i \in I} (z_i = u_i) \vee \mathcal{R}'$, the relation $\mathcal{P}' \leq \mathcal{P}''$ holds, since the omission of parts of the variable bindings in the mgu may not decrease the set of solutions. But then the equations $z_i = u_i$ may be deleted by the U_4-rule recalled in Section 2.2. Thus, $\mathcal{P}''$ is equivalent to $(\forall \vec{y})\mathcal{R}'$. Hence, the equivalence $\mathcal{P} \approx (\forall \vec{y})\mathcal{R}'$ follows from the relations $\mathcal{P} \approx \mathcal{P}' \leq \mathcal{P}'' \approx (\forall \vec{y})\mathcal{R}' \leq \mathcal{P}$. □

Note that Case 3 in the above lemma is the only place in our transformation into PFE-form where the restriction to an infinite domain is essential. In other words, the ultimate reason why equational problems in CNF over a finite domain are one level higher in the polynomial hierarchy than in the case of an infinite domain, is that the U_4-rule recalled in Section 2.2 only holds in the case of an infinite domain.

The following lemma allows the elimination of the remaining parameters from the equations. Note that again there is no appropriate transformation rule of Comon and Lescanne (1989) available for this purpose: eliminating these parameters via the explosion rule E has exponential time complexity. And the U_2-rule can only be used to eliminate

the parameters from the equations if all disequations $x_{i_j} \neq t_j$ have been broken down to disequations between variables only. Again the necessary applications of the explosion rule require exponential time.

Lemma 5.3 (Elimination of the Remaining Parameters from the Equations). *Let D be an arbitrary domain and let $\mathcal{P} \equiv (\forall \vec{y})(\mathcal{E} \vee \vec{z} \neq \vec{t})$ be an equational problem over D with free variables in $\vec{x}$, such that $\mathcal{E} \equiv s_1 = t_1 \vee \cdots \vee s_k = t_k$ is a disjunction of equations, $\vec{z} \subseteq \vec{x}$ is a vector of free variables which do not occur in the term tuple $\vec{t}$ and all parameters in $\vec{y}$ actually do occur in $\vec{t}$. Then $\mathcal{P}$ is equivalent to $\mathcal{P}' \equiv [(\exists \vec{y})(\vec{z} = \vec{t} \wedge \mathcal{E})] \vee [(\forall \vec{y})\vec{z} \neq \vec{t}]$.*

Proof. The equivalence $\mathcal{P} \approx (\forall \vec{y})[(\vec{z} = \vec{t} \wedge \mathcal{E}) \vee \vec{z} \neq \vec{t}]$ is due to the equivalence $[A \vee B] \approx [(\neg B \wedge A) \vee B]$, which holds for any logical formulae A and B. Replacing the universal quantifier for some variables by an existential one cannot decrease the set of solutions. Hence, the relation $\mathcal{P} \leq \mathcal{P}'$ clearly holds. It therefore only remains to show that every solution of $\mathcal{P}'$ is also a solution of $\mathcal{P}$. Let the ground substitution σ on the free variables $\vec{x}$ be a solution of $\mathcal{P}'$. If σ is a solution of $(\forall \vec{y})\vec{z} \neq \vec{t}$, then it is of course also a solution of $\mathcal{P} \equiv (\forall \vec{y})(\mathcal{E} \vee \vec{z} \neq \vec{t})$. So let σ be a solution of $(\exists \vec{y})(\vec{z} = \vec{t} \wedge \mathcal{E})$ and let τ be an arbitrary ground substitution for the variables $\vec{y}$. We have to show that then $(\mathcal{E} \vee \vec{z} \neq \vec{t})(\sigma \cup \tau) \approx \top$ holds:

If $(\vec{z} \neq \vec{t})(\sigma \cup \tau) \approx \top$ holds, then $(\mathcal{E} \vee \vec{z} \neq \vec{t})(\sigma \cup \tau) \approx \top$ clearly holds. Hence, we only have to prove the equivalence $\mathcal{E}(\sigma \cup \tau) \approx \top$ for the case where $(\vec{z} \neq \vec{t})(\sigma \cup \tau) \approx \bot$ holds.

By assumption, σ is a solution of $(\exists \vec{y})(\vec{z} = \vec{t} \wedge \mathcal{E})$. Therefore, there exists a ground substitution φ for the variables $\vec{y}$ such that $(\vec{z} = \vec{t} \wedge \mathcal{E})(\sigma \cup \varphi) \approx \top$. In particular, $\mathcal{E}(\sigma \cup \varphi) \approx \top$ holds. It is, therefore, sufficient to show that τ and φ are identical: remember that we are dealing with the case where $(\vec{z} \neq \vec{t})(\sigma \cup \tau) \approx \bot$ and, therefore, $(\vec{z} = \vec{t})(\sigma \cup \tau) \approx \top$ holds, i.e. $\sigma \cup \tau$ is a unifier of $\vec{z} = \vec{t}$. Furthermore, $(\vec{z} = \vec{t} \wedge \mathcal{E})(\sigma \cup \varphi) \approx \top$ holds by the definition of φ and, therefore, $\sigma \cup \varphi$ is also a unifier of $\vec{z} = \vec{t}$. By assumption, all parameters y_i actually do occur in $\vec{t}$. Hence, for every unifier $(\sigma \cup \eta)$ of $\vec{z} = \vec{t}$, where σ is a ground substitution on the variables $\vec{x}$, the extension η to the variables $\vec{y}$ is uniquely determined by σ. But then, φ and τ are indeed identical. □

Note that the conjunction $(\vec{z} = \vec{t} \wedge \mathcal{E})$ in the above lemma with $\mathcal{E} \equiv s_1 = t_1 \vee \cdots \vee s_k = t_k$ can be represented as a disjunction $\mathcal{E}'$ of equations (of term tuples), namely: $\mathcal{E}' \equiv (\vec{z}, s_1) = (\vec{t}, t_1) \vee \cdots \vee (\vec{z}, s_k) = (\vec{t}, t_k)$. Hence, the only part missing in our transformation into the PFE-form is the elimination of the existentially quantified variables from the right-hand side of the disequations. But this can be easily achieved by applying the U_2-rule recalled in Section 2.2 in the opposite direction:

Lemma 5.4. (Elimination of the Existentially Quantified Variables from the Right-hand Side of the Disequations). *Let D be an arbitrary domain and let $\mathcal{P} \equiv (\forall \vec{y})\vec{w} \neq \vec{s}$ be an equational problem over D such that $\vec{w}$ is a vector of free variables. Moreover, let $\vec{u} = (u_1, \ldots, u_l)$ denote the free variables occurring in $\vec{s}$ and suppose that $\vec{u} \cap \vec{w} = \emptyset$ holds.*

Then $\mathcal{P}$ is equivalent to $\mathcal{P}' \equiv (\forall \vec{y})(\forall \vec{z})(\vec{w}, \vec{u}) \neq (\vec{s}\eta, \vec{z})$, where $\vec{z} = (z_1, \ldots, z_l)$ is a vector of fresh, pairwise distinct variables and the substitution η is defined as $\eta = \{u_1 \leftarrow z_1, \ldots, u_l \leftarrow z_l\}$.

By combining the transformation steps from the Lemmas 5.1 through 5.4, we get the following theorem on the transformation into PFE-form:

Theorem 5.1 (Transformation into PFE-form). *Let D be an infinite domain. Then the transformation of equational problems over D from CNF into PFE-form can be done in polynomial time.*

Proof. Let $\mathcal{P} \equiv \exists\vec{z}\forall\vec{y}[\mathcal{P}_1 \wedge \cdots \wedge \mathcal{P}_n]$ be an equational problem, such that the $\mathcal{P}_i$'s are quantifier-free disjunctions of equations and disequations. Of course, the universal quantifiers may be shifted in front of the $\mathcal{P}_i$' s, thus yielding an equivalent problem $\mathcal{P}' \equiv \exists\vec{z}\,[\forall\vec{y}\mathcal{P}_1] \wedge \cdots \wedge [\forall\vec{y}\mathcal{P}_n]$. It is then possible, to apply the transformation steps from Lemmas 5.1 through 5.4 to every conjunct $[\forall\vec{y}\mathcal{P}_i]$:

By Lemma 5.1, the disequations can be simplified in such a way, that the left-hand sides of the disequations consist of pairwise distinct, existentially quantified variables, which do not occur on the right-hand side of any disequation. Lemma 5.2 can be used to eliminate all parameters from the equations that do not occur in the disequations. The remaining parameters in the equations can then be eliminated by means of Lemma 5.3. When the equations contain no more parameters, the universal quantifiers can be shifted in front of the disequations. The universally quantified disequations thus produced can be brought into the desired form via Lemma 5.4. Finally, by appropriately renaming the existentially quantified variables introduced by Lemma 5.3 and shifting the existential quantifiers to the front of the formula, the transformation into PFE-form is finished.

The *correctness* of these transformation steps has been proven in Lemmas 5.1 through 5.4. As for the *complexity*, note that the only operations required for this transformation basically are the computation of several most general unifiers, the application of these mgu's to the appropriate terms and checking whether certain variables occur in the resulting terms or in the unifiers themselves, respectively. Hence, all of these steps can be done in polynomial time, provided that unifiers are treated as directed acyclic graphs rather than as strings. □

In the following example we put this transformation to work. It should thus help to illustrate the main ideas of Theorem 5.1 and the preceding lemmas:

Example 5.1. Let H denote the Herbrand universe with signature $\Sigma = \{a, f, g\}$ and let $\mathcal{P} \equiv \exists(z_1, z_2, z_3, z_4)\,\forall(y_1, y_2, y_3, y_4)[\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \mathcal{P}_3]$ be an equational problem over H, where the $\mathcal{P}_i$'s are defined as follows:

$$
\begin{aligned}
\mathcal{P}_1 \;\equiv\;& f(z_1, g(z_2)) = f(y_1, z_3) \vee \\
& g(y_1) = z_3 \vee \\
& f(a, y_2) = f(z_2, g(y_4)) \vee \\
& f(g(y_2), z_1) \neq f(y_3, g(y_1)) \vee \\
& g(z_3) \neq z_2 \\
\mathcal{P}_2 \;\equiv\;& f(y_1, a) = f(g(z_2), a) \vee \\
& f(g(z_4), y_1) \neq f(z_2, a) \vee \\
& g(y_1) \neq g(g(y_3))
\end{aligned}
$$

$$\begin{aligned}\mathcal{P}_3 \equiv\ & f(g(z_2), z_1) = f(g(y_1), y_1) \vee \\ & g(y_2) = y_3 \vee \\ & f(a, y_4) = f(z_2, g(y_2)) \vee \\ & f(f(a, y_3), g(y_2)) \neq f(z_1, y_4) \vee \\ & g(z_2) \neq g(f(a, z_3)) \vee \\ & z_4 \neq y_1.\end{aligned}$$

After shifting the universal quantifiers in front of the $\mathcal{P}_i$' s, we can apply the transformation steps from Lemmas 5.1 through 5.4 to the resulting subproblems $[\forall(y_1, y_2, y_3, y_4)\mathcal{P}_i]$: note that no unifier exists for the negated disequations in $\mathcal{P}_2$, while the negated disequations of the other subproblems are unifiable with the mgu's $\vartheta_1 = \{z_1 \leftarrow g(y_1), y_3 \leftarrow g(y_2), z_2 \leftarrow g(z_3)\}$ and $\vartheta_3 = \{z_1 \leftarrow f(a, y_3), z_2 \leftarrow f(a, z_3), y_4 \leftarrow g(y_2), y_1 \leftarrow z_4\}$, respectively. Hence, by Lemma 5.1, the $\mathcal{P}_i$'s may be transformed as follows:

$$\begin{aligned}\mathcal{P}_1^{(1)} \equiv\ & f(g(y_1), g(g(z_3))) = f(y_1, z_3) \vee \\ & g(y_1) = z_3 \vee \\ & f(a, y_2) = f(g(z_3), g(y_4)) \vee \\ & z_1 \neq g(y_1) \vee \\ & z_2 \neq g(z_3) \\ \mathcal{P}_2^{(1)} \equiv\ & \top \\ \mathcal{P}_3^{(1)} \equiv\ & f(g(f(a, z_3)), f(a, y_3)) = f(g(z_4), z_4) \vee \\ & g(y_2) = y_3 \vee \\ & f(a, g(y_2)) = f(z_2, g(y_2)) \vee \\ & z_1 \neq f(a, y_3) \vee \\ & z_2 \neq f(a, z_3).\end{aligned}$$

Let e_{ij} denote the jth equation in the problem $\mathcal{P}_i^{(1)}$, i.e.

$$\begin{aligned} & e_{11} \equiv f(g(y_1), g(g(z_3))) = f(y_1, z_3) && e_{31} \equiv f(g(f(a, z_3)), f(a, y_3)) \\ & && \quad = f(g(z_4), z_4) \\ & e_{12} \equiv g(y_1) = z_3 && e_{32} \equiv g(y_2) = y_3 \\ & e_{13} \equiv f(a, y_2) = f(g(z_3), g(y_4)) && e_{33} \equiv f(a, g(y_2)) = f(z_2, g(y_2)).\end{aligned}$$

Now let ϑ_{ij} denote the mgu of e_{ij}. Then the following mgu's exist:

$$\begin{aligned} & \vartheta_{12} = \{z_3 \leftarrow g(y_1)\} && \vartheta_{31} = \{z_4 \leftarrow f(a, y_3), z_3 \leftarrow y_3\} \\ & \vartheta_{32} = \{y_3 \leftarrow g(y_2)\} && \vartheta_{33} = \{z_2 \leftarrow a\}.\end{aligned}$$

By Lemma 5.2, the equations e_{11} and e_{13} may be deleted, since they are not unifiable. Likewise, the equation e_{32} may be deleted, since ϑ_{32} contains the parameter y_2, which does not occur in the disequations. Finally, in equation e_{33}, we may replace y_2 by the constant symbol a. We thus get the following problems:

$$\begin{aligned}\mathcal{P}_1^{(2)} \equiv\ & g(y_1) = z_3 \vee \\ & z_1 \neq g(y_1) \vee \\ & z_2 \neq g(z_3)\end{aligned}$$

$$\begin{aligned}\mathcal{P}_3^{(2)} \equiv\ & f(g(f(a,z_3)),\ f(a,y_3)) = f(g(z_4),z_4)\vee\\ & f(a,g(a)) = f(z_2,g(a))\vee\\ & z_1 \neq f(a,y_3)\vee\\ & z_2 \neq f(a,z_3).\end{aligned}$$

By Lemma 5.3, the problems $[\forall(y_1,y_2,y_3,y_4)\mathcal{P}_i^{(2)}]$ can be further transformed into the following problems $\mathcal{P}_i^{(3)}$ (Note that, for simplicity, we omit all variables y_j from the universal quantifier prefix that have already been eliminated from the corresponding formula. Moreover, we use term tuples as a short-hand notation for conjunctions of equations and disjunctions of disequations, respectively):

$$\begin{aligned}\mathcal{P}_1^{(3)} \equiv\ & (\exists y_1)(z_1,z_2,g(y_1)) = (g(y_1),g(z_3),z_3)\vee\\ & (\forall y_1)(z_1,z_2) \neq (g(y_1),g(z_3))\\ \mathcal{P}_3^{(3)} \equiv\ & (\exists y_3)(z_1,z_2,\ f(g(f(a,z_3)),\ f(a,y_3))) = (f(a,y_3),\ f(a,z_3),\\ & f(g(z_4),z_4))\vee\\ & (z_1,z_2,\ f(a,g(a))) = (f(a,y_3),\ f(a,z_3),\ f(z_2,g(a)))\vee\\ & (\forall y_3)(z_1,z_2) \neq (f(a,y_3),\ f(a,z_3)).\end{aligned}$$

It is now possible to shift the universal quantifiers in front of the disequations. We can then apply Lemma 5.4 to get the following equational problems:

$$\begin{aligned}\mathcal{P}_1^{(4)} \equiv\ & (\exists y_1)(z_1,z_2,g(y_1)) = (g(y_1),g(z_3),z_3)\vee\\ & (\forall y_1,v)(z_1,z_2,z_3) \neq (g(y_1),g(v),v)\\ \mathcal{P}_3^{(4)} \equiv\ & (\exists y_3)(z_1,z_2,\ f(g(f(a,z_3)),\ f(a,y_3))) = (f(a,y_3),\ f(a,z_3),\\ & f(g(z_4),z_4))\vee\\ & (z_1,z_2,\ f(a,g(a))) = (f(a,y_3),\ f(a,z_3),\ f(z_2,g(a)))\vee\\ & (\forall y_3,v)(z_1,z_2,z_3) \neq (f(a,y_3),\ f(a,v),v).\end{aligned}$$

After renaming the existentially quantified variables y_1 in $\mathcal{P}_1^{(4)}$ and y_3 in $\mathcal{P}_3^{(4)}$ by the fresh variables u_1 and u_2, respectively, we can shift all existential quantifiers to the front of the conjunction $\mathcal{P}_1^{(4)}\wedge\mathcal{P}_3^{(4)}$. We thus get the formula $\mathcal{P}' \equiv \exists(z_1,z_2,z_3,z_4,u_1,u_2)[\mathcal{P}_1^{(5)}\wedge\mathcal{P}_3^{(5)}]$ with

$$\begin{aligned}\mathcal{P}_1^{(5)} \equiv\ & (z_1,z_2,g(u_1)) = (g(u_1),g(z_3),z_3)\vee\\ & (\forall y_1,v)(z_1,z_2,z_3) \neq (g(y_1),g(v),v)\\ \mathcal{P}_3^{(5)} \equiv\ & (z_1,z_2,\ f(g(f(a,z_3)),\ f(a,u_2))) = (f(a,y_3),\ f(a,z_3),\ f(g(z_4),z_4))\vee\\ & (z_1,z_2,\ f(a,g(a))) = (f(a,u_2),\ f(a,z_3),\ f(z_2,g(a)))\vee\\ & (\forall y_3,v)(z_1,z_2,z_3) \neq (f(a,y_3),\ f(a,v),v).\end{aligned}$$

Then $\mathcal{P}'$ is the desired equational formula in PFE-form.

5.2. NP-membership

The transformation from CNF into PFE-form in Section 5.1 forms the basis of our NP-membership proof for the satisfiability problem of equational problems in CNF. In fact, by the polynomial time complexity of this transformation, we can restrict ourselves without loss of generality to the case of equational problems in PFE-form. At the heart

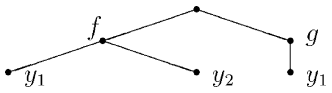

Fig. 3. Tree representation of $(f(y_1, y_2), g(y_1))$.

of our NP-membership proof will be an idea of Gottlob and Pichler (1999), where an appropriate representation of the complement of a term tuple $\vec{t}$ (with respect to a given Herbrand universe H) is provided, i.e. let $\vec{t}$ be a k-tuple of (in general, non-ground) terms over H. Then the set of all ground term tuples $\vec{s} \in H^k$ that are not instances of $\vec{t}$ is referred to as the complement of $\vec{t}$.

Recall that every term tuple can be represented as a labelled tree, namely: the root node has no label and the degree of the root corresponds to the dimension of the tuple. All other internal nodes are labelled with proper function symbols and the degree of these nodes corresponds to the arity of the labelling symbol. Finally, the leaf nodes are either labelled by a constant or a variable symbol.

Then the tuples $\vec{s}$ from the complement of $\vec{t}$ can be obtained as follows: consider the tree representation T of $\vec{t}$, "deviate" from T at some node and close all other branches of T as early as possible with new, pairwise distinct variables. The result of such a deviation is denoted as $\text{dev}_{(p,q)}(\vec{t})$: intuitively, p indicates the position *where* one deviates from $\vec{t}$ and q tells us *how* one deviates:

- If p refers to a node in T that is labelled by some *function symbol* f (constants are considered as function symbols of arity 0), then $\text{dev}_{(p,q)}(\vec{t})$ is defined, iff q is a function symbol different from f with arity $\alpha \geq 0$. In this case, $\text{dev}_{(p,q)}(\vec{t})$ is constructed from $\vec{t}$ by replacing the subterm $[\vec{t}\,|\,p]$ at position p by the term $q(z_1, \ldots, z_\alpha)$ and by closing all other branches of T as early as possible with new variables.
- If the node in T corresponding to the position p is labelled by a *variable* y which *occurs somewhere else* in $\vec{t}$, then $\text{dev}_{(p,q)}(\vec{t})$ is defined, iff q is such a position in $\vec{t}$ where y also occurs. Then $\text{dev}_{(p,q)}(\vec{t})$ is constructed from $\vec{t}$ by replacing y at position p with a fresh variable z and by restricting the ground instances of the resulting term tuple through the constraint $z \neq y$. Again, all other branches are closed as early as possible with new variables.
- If the node in the tree representation T of $\vec{t}$ corresponding to the position p is labelled by a *variable* that *occurs only once* in $\vec{t}$, then no deviation at all is possible at position p and, therefore, $\text{dev}_{(p,q)}(\vec{t})$ is undefined for every q.

A formal definition of this idea and the proof that the complement of a term tuple $\vec{t}$ is actually captured by these deviations at nodes of the tree representation of $\vec{t}$ can be found in Gottlob and Pichler (1999). This idea is illustrated in the following example:

Example 5.2. Let H denote the Herbrand universe with signature $\Sigma = \{f, g, a\}$ and let $\vec{t} = (f(y_1, y_2), g(y_1))$ be a term tuple over H.

Then the tree corresponding to $\vec{t}$ is depicted in Fig. 3. Note that no deviation is possible at position 1.2, since $y_2 = [\vec{t}\,|1.2]$ is a variable occurring only once in $\vec{t}$. For all other

positions p in $\vec{t}$, $\mathrm{dev}_{(p,q)}(\vec{t})$ is defined for appropriately chosen q. Hence, every ground term tuple $\vec{s}$ from the complement of $\vec{t}$ is an instance of one of the following (possibly constrained) tuples:

$$\begin{array}{ll} \mathrm{dev}_{(1,a)} = (a, v) & \mathrm{dev}_{(2,f)} = (v,\ f(z_1, z_2)) \\ \mathrm{dev}_{(1,g)} = (g(z), v) & \mathrm{dev}_{(1.1,2.1)} = [(f(z, y_2), g(y_1)) : z \neq y_1] \\ \mathrm{dev}_{(2,a)} = (v, a) & \mathrm{dev}_{(2.1,1.1)} = [(f(y_1, y_2), g(z)) : z \neq y_1]. \end{array}$$

This idea of representing the complement of a term tuple by a set of (possibly constrained) tuples can be used directly for eliminating all parameters from an equational problem in PFE-form as the following example illustrates.

Example 5.3. Let H again denote the Herbrand universe with signature $\Sigma = \{f, g, a\}$ and let $\mathcal{P} \equiv (\forall \vec{y})(x_1, x_2) \neq \vec{t}$ be an equational problem over H with $\vec{t} = (f(y_1, y_2), g(y_1))$. Then the representation of the complement of $\vec{t}$ in Example 5.2 yields the following parameter-free problem $\mathcal{P}'$, which is equivalent to $\mathcal{P}$:

$$\begin{array}{rl} \mathcal{P}' \ \equiv\ & (\exists v)(x_1, x_2) = (a, v) \vee \\ & (\exists v)(\exists z)(x_1, x_2) = (g(z), v) \vee \\ & (\exists v)(x_1, x_2) = (v, a) \vee \\ & (\exists v)(\exists z_1)(\exists z_2)(x_1, x_2) = (v,\ f(z_1, z_2)) \vee \\ & (\exists z)(\exists y_1)(\exists y_2)[(x_1, x_2) = (f(z, y_2), g(y_1)) \wedge z \neq y_1] \vee \\ & (\exists z)(\exists y_1)(\exists y_2)[(x_1, x_2) = (f(y_1, y_2), g(z)) \wedge z \neq y_1]. \end{array}$$

With this transformation of a universally quantified disequation into a parameter-free disjunction, we are ready to prove the following NP-membership result:

Lemma 5.5 (NP-membership of PFE-form). *Let D be an infinite domain. Then the satisfiability problem for equational problems in PFE-form over D is in NP.*

Proof. Let $\mathcal{P} \equiv (\exists \vec{x})[(e_{11} \vee \cdots \vee e_{1k_1}) \vee (\forall \vec{y}_1)(\vec{z}_1 \neq \vec{t}_1)] \wedge \cdots \wedge [(e_{n1} \vee \cdots \vee e_{nk_n}) \vee (\forall \vec{y}_n)(\vec{z}_n \neq \vec{t}_n)]$ be an equational problem such that the e_{ij}'s are equations, the $\vec{z}_i$'s are tuples of existentially quantified variables from $\vec{x}$ and the $\vec{t}_i$'s are term tuples with variables in $\vec{y}_i$. Then the following non-deterministic algorithm checks in polynomial time that $\mathcal{P}$ is satisfiable.

1. For every $i \in \{1, \ldots, n\}$, either guess an equation $e_{i\alpha_i}$ in the ith disjunction with $\alpha_i \in \{1, \ldots, k_i\}$ or a (possibly constrained) tuple $\mathrm{dev}_{(p_i,q_i)}(\vec{t}_i)$ from the complement of $\vec{t}_i$.
2. Define the conjunction $\mathcal{Q} \equiv (\exists \vec{x})[\mathcal{Q}_1 \wedge \cdots \wedge \mathcal{Q}_n]$ of equations and disequations in the following way: if in the first step an equation $e_{i\alpha_i}$ was guessed, then $\mathcal{Q}_i \equiv e_{i\alpha_i}$. If a tuple $\vec{s}_i$ without constraints from the complement of $\vec{t}_i$ was guessed, then $\mathcal{Q}_i \equiv (\exists \vec{w}_i)\vec{z}_i = \vec{s}_i$, where $\vec{w}_i$ denotes the variables in $\vec{s}_i$. Finally, if a constrained tuple $[\vec{s}_i : u \neq v]$ from the complement of $\vec{t}_i$ was guessed, then $\mathcal{Q}_i \equiv (\exists \vec{w}_i)(\vec{z}_i = \vec{s}_i \wedge u \neq v)$, where $\vec{w}_i$ again denotes the variables in $\vec{s}_i$.
3. Rename all variables in $\mathcal{Q}$ appropriately apart such that the existential quantifiers may be shifted in front of the conjunction.
4. Check the satisfiability of the resulting conjunction of equations and disequations via the test in Lemma 3.1, i.e. check that the mgu ϑ of the equations exists and that

the application of ϑ to the disequations does not produce a trivial disequation of the form $\vec{t} \neq \vec{t}$.

The non-deterministic guess of a disjunct in Step 1 corresponds to the distributivity of $\vee$ and $\wedge$. The variable renaming and quantifier shifting in Step 3 is not problematical at all and the satisfiability test for a parameter-free conjunction of equations and disequations in Step 4 has already been discussed in Lemma 3.1. Hence, the only critical part for the *correctness* of the above algorithm is Step 2. But the correctness of the representation of the complement has been proven in Gottlob and Pichler (1999) and the correctness of the transformation of a disequation $(\forall \vec{y}_i)(\vec{z}_i \neq \vec{t}_i)$ into a parameter-free disjunction follows easily.

The crucial point for the *polynomial time complexity* of the algorithm is that the size of the terms involved in the complement of a tuple $\vec{t}_i$ depends polynomially on the size of an input problem $\mathcal{P}$ even if the terms in $\mathcal{P}$ are represented as dags (=directed acyclic graphs). This is due to the fact that $\mathrm{dev}_{(p,q)}(\vec{t}_i)$ is either defined around one path (if p is a non-variable position) or two paths (if p is a variable position) of $\vec{t}_i$, i.e. the number of positions (which corresponds to the number of nodes in the tree representation) in the terms occurring in $\mathrm{dev}_{(p,q)}(\vec{t}_i)$ is linearly bounded in the term depth of $\vec{t}_i$ (where the multiplicative constant is basically the maximum arity of the function symbols in D). Hence, even if the string representation of terms represented by dags may have exponential size, the size of the string representation of the (possibly constrained) tuple $\mathrm{dev}_{(p_i,q_i)}(\vec{t}_i)$ is polynomially bounded with respect to the dag representation of $\vec{t}_i$. Moreover, the satisfiability test in Lemma 3.1 can be done in polynomial time independently of the chosen term representation. Therefore, the overall complexity of this non-deterministic algorithm is polynomial. □

Note that the restriction to an infinite domain in Lemma 5.5 is not essential. It was only done for convenience since, analogously to the proof of Lemma 3.1, we would have had to treat the case of a finite domain and an infinite domain separately. Moreover, the PFE-form is only interesting for an infinite domain since the polynomial time transformation in Section 5.1 works only in this case. The following theorem follows immediately from Theorem 5.1 and Lemma 5.5:

Theorem 5.2 (CNF Over an Infinite Domain). *Let D be an infinite domain. Then the satisfiability problem for equational problems in CNF over D is NP-complete.*

6. Algorithms for solving equational problems

As usual, the complexity analysis of a given problem is not the end of the story. In general, one will try to apply the theoretical insight into the inherent complexity of a problem to the construction of new and more efficient algorithms.

An important conclusion to be drawn from the complexity results in Section 4 is that in the case of a **finite domain**, the transformation from arbitrary form or DNF into CNF via the distributivity of $\wedge$ and $\vee$ does not make sense at all. Hence, when searching for a more efficient algorithm, one should actually try to take the arbitrary form or DNF itself as the starting point for a satisfiability test. Note that this is in great contrast to the case of an

infinite domain, where the transformation into CNF via the distributivity of $\wedge$ and $\vee$ (either as a preprocessing step as in Comon and Lescanne, 1989 or gradually as in Comon and Delor, 1994) is clearly a reasonable strategy. In order to illustrate this point in some more detail, we revisit the algorithm of Comon and Lescanne (1989) for deciding equational problems over a finite or an infinite domain, respectively. In particular, we shall sketch how this algorithm can be improved in the case of a finite domain.

The goal of the algorithm of Comon and Lescanne (1989) is a transformation of a given equational problem $\mathcal{P}$ into the so-called *definition with constraints* form (DWC-form, for short), which is either the trivially true problem $\top$ or the trivially false problem $\bot$ or a purely existentially quantified equational problem in DNF, where the disjuncts are of the form $\mathcal{P}_i \equiv [x_{i1} = s_{i1} \wedge \cdots \wedge x_{ik_i} = s_{ik_i} \wedge z_{i1} \neq t_{i1} \wedge \cdots \wedge z_{il_i} \neq t_{il_i}]$, such that the x_{ij}'s are variables that occur exactly once in $\mathcal{P}_i$ and every z_{ij} is a variable that is syntactically different from the term t_{ij} on the right-hand side. Moreover, disequations are only allowed in the case of an infinite domain. It is shown in Comon and Lescanne (1989), that an *arbitrary* equational problem is satisfiable, iff it can be transformed into a DWC-form that is syntactically different from $\bot$.

Now suppose that we have to decide the satisfiability of an arbitrary equational problem $\mathcal{P}$. Then the transformation of Comon and Lescanne (1989) looks as follows:

$$\exists^* \forall^* \text{ form} \overset{(1)}{\Longrightarrow} \exists^* \forall^*\text{-CNF} \overset{(2)}{\Longrightarrow} \exists^*\text{-CNF} \overset{(3)}{\Longrightarrow} \exists^*\text{-DNF} \overset{(4)}{\Longrightarrow} \text{DWC-form}.$$

In the case of an infinite domain, this strategy clearly makes sense: step (1) has, in general, exponential complexity. Steps (2) and (3) correspond to the guessing in our NP-algorithm in Section 5. Hence, these two steps together also have exponential worst-case complexity. Finally, step (4) is rather cheap since it only consists of unification. We thus end up with two orthogonal sources of exponential complexity, which were somehow to be expected by the Σ_2^p-hardness of equational problems in arbitrary form over any non-trivial domain (cf. Theorem 4.1). Note that the algorithm of Comon and Lescanne (1989) was refined in Comon and Delor (1994). In particular, in Comon and Delor (1994), the expensive distributivity of $\vee$ and $\wedge$ is applied only when it is actually required rather than as a preprocessing step. However, in the worst case, the two above-mentioned sources of exponential complexity are still present.

On the other hand, in the case of a finite domain, all of steps (1) through (4) have exponential complexity. As far as steps (1) and (3) are concerned, this is clear. Steps (2) and (4) also have exponential complexity due to the expensive transformation rules U_5 and E, respectively, recalled in Section 2.2. Let $\Sigma = \{a_1, \ldots, a_K\}$ denote a signature without proper function symbols and suppose that our equational problem is of the form $(\exists \vec{x})(\forall \vec{y})[P \wedge Q]$, such that Q contains a universally quantified variable y from $\vec{y}$. Then, by the rule U_5, we may replace $P \wedge Q$ by $[P \wedge Q(y \leftarrow a_1) \wedge \cdots \wedge Q(y \leftarrow a_K)]$. Likewise, let us consider an equational problem $(\exists \vec{x})[P_1 \vee \cdots \vee P_n]$ in $\exists^*$-DNF, such that the disjunct P_i is of the form $P_i \equiv Q \wedge x \neq t \wedge R$. Then P_i may be transformed by the rule E into the disjunction $P_i' \equiv (Q \wedge a_1 \neq t \wedge R) \vee \cdots \vee (Q \wedge a_K \neq t \wedge R)$.

Of course, the transformation system given in Comon and Lescanne (1989) also contains cheap rules, whose application is always preferred to the expensive ones recalled

above. However, in general, the application of the expensive rules cannot be avoided and the overall complexity of steps (2) and (4) is thus exponential in the worst case.

In other words, the exponential complexity of steps (1) and (3) is not justified in the case of a finite domain since, in contrast to an infinite domain, the resulting equational problems are just as hard to solve as the original ones. Instead we should eliminate straight away the universally quantified variables via the rule U_5. Note that this rule is not restricted to a particular form of the equational problem. Moreover, the rule E does not necessarily have to be restricted to the elimination of non-ground disequations. Instead, it can be used to eliminate all existentially quantified variables, i.e. let $(\exists\vec{x})[P \vee Q]$ be an equational problem, such that Q contains an existentially quantified variable x from $\vec{x}$. Then, analogously to the rule U_5, we may replace $P \vee Q$ by $[P \vee Q(x \leftarrow a_1) \vee \cdots \vee Q(x \leftarrow a_K)]$. Of course, both the elimination of the universally quantified variables via the rule U_5 and the elimination of the existentially quantified variables via the (modified) rule E have exponential complexity. However, by the Σ_2^p-completeness result in Theorem 4.1, we cannot really expect to get rid of these two orthogonal sources of non-polynomial complexity.

In the case of an **infinite domain** , our NP-membership proof in Section 5 can be viewed as a possible step towards a more efficient algorithm. Recall that the transformations in Lemmas 5.1 through 5.4 mainly consist of unification steps. Hence, the whole transformation into PFE-form is rather cheap and, as the NP-membership result in Lemma 5.5 illustrates, quite useful. As has already been mentioned before, no polynomial time transformation into PFE-form was provided by previous works in this area (cf., in particular, Comon and Lescanne, 1989 and Comon and Delor, 1994).

For further improvements, we recall the so-called TTC problem (=*term tuple cover problem*). For an arbitrary Herbrand universe H, it is defined as follows:

TTC Problem

Input: A set $M = \{\vec{t}_1, \ldots, \vec{t}_n\}$ of k-tuples of terms over H.

Question: Is every ground term tuple $\vec{s} \in H^k$ an instance of some tuple $\vec{t}_i \in M$?

The complementary problem coTTC can be easily translated into the satisfiability problem of equational problems of the following form:

Definition (coTTC-form). An equational problem $\mathcal{P}$ is said to be in *coTTC-form*, iff it is of the form $\mathcal{P} \equiv (\exists\vec{x})[(\forall\vec{y}_1)(\vec{x} \neq \vec{t}_1) \wedge \cdots \wedge (\forall\vec{y}_n)(\vec{x} \neq \vec{t}_n)]$, such that $\vec{t}_i = (t_{i1}, \ldots, t_{ik})$ is a term tuple with variables in $\vec{y}_i$ for every $i \in \{1, \ldots, n\}$.

Note that the transformation into PFE-form given in Section 5.1 can be easily extended to a transformation into a collection of equational problems in coTTC-form. It is thus possible to apply efficient algorithms for the well-studied term tuple cover problem to equational problems. Actually, the algorithm of Comon and Lescanne (1989) as well as a deterministic version of our NP-algorithm in Theorem 5.1 followed by Lemma 5.5 have exponential complexity with respect to the size of the terms involved (in particular, with respect to the term depth). In contrast, a satisfiability test for equational problems in coTTC-form analogously to the term tuple cover algorithm of Pichler (2000b) is

exponential in the total number of equations and disequations, while the size of the terms only has polynomial influence on the overall complexity.

7. Conclusions and future work

The main results of this paper are the NP-completeness (and, in particular, the NP-membership) of the satisfiability problem for equational problems in CNF over an infinite domain and the Σ_2^p-completeness in the case of CNF over a finite domain. For equational problems in DNF, the Σ_2^p-hardness was easily established for any non-trivial domain. However, the Σ_2^p-membership for equational problems in DNF has only been shown in the case of a finite domain. The obvious upper bound on the complexity of equational problems in DNF over an infinite domain is NEXPTIME, since we can of course first transform the DNF into CNF via the distributivity of $\wedge$ and $\vee$ (in general, at the expense of an exponential blow-up) and then apply the NP-algorithm from Section 5. Closing the gap between the Σ_2^p-lower bound and the NEXPTIME-upper bound is an interesting open problem for future research in this area.

Recall that we have only considered the case where all terms (and, in particular, all variables) in an equational problem are interpreted over the same domain. One should now try to extend the results obtained here to the case of many sorts. Actually, it seems as though this extension is not too difficult. After all, it has turned out in our investigations that we only have to be careful whether a domain is finite or infinite. Nevertheless, the details of such an extension to many sorts have to be worked out yet.

More importantly, the search for more efficient algorithms on equational problems should be continued. A major lesson to be learnt from our complexity results is that—in contrast to the algorithm of Comon and Lescanne (1989) —one should not try to treat the cases of a finite domain and of an infinite domain, respectively, in a uniform way. In Section 6, we have shown how simple modifications of the algorithm of Comon and Lescanne (1989) may lead to a significant improvement in the case of a finite domain. Likewise, some ideas for an improvement in the case of an infinite domain have been mentioned. Searching for further improvements both in the case of a finite domain and an infinite domain is an important goal for future research.

In this paper we have concentrated on equational formulae with $\exists^* \forall^*$-prefix. This restriction is somehow justified since, in many important applications, equational formulae of this form occur in a natural way. Nevertheless, an extension of our algorithm to arbitrary equational formulae would be desirable. In Comon and Delor (1994), an algorithm is presented which neither requires a CNF nor a specific quantifier prefix. Instead, the expensive distributivity rules are only applied when this is really necessary. Moreover, a whole collection of rules dealing with single quantifiers and combinations of quantifiers are provided. Of course, by the high inherent complexity of equational formulae with no restriction on the quantifier occurrences (i.e. non-elementary complexity in the case of an infinite domain and PSPACE-completeness in the case of a finite domain, cf. Vorobyov, 1996 and Kunen, 1987, respectively), there is a clear limit up to which the worst case complexity can possibly be improved. Nevertheless, the ideas of Comon and Delor (1994) may lead to significant improvements in many cases. One should now try

to combine the ideas from Section 5 with the algorithm of Comon and Delor (1994). In particular, integrating our transformation rules from Lemmas 5.1 through 5.4 into the rule system from Comon and Delor (1994) and applying these new rules before any expensive transformation rule (like the explosion rule) is applied, may possibly increase the efficiency of the algorithm of Comon and Delor (1994). But still, a thorough complexity analysis of the algorithm of Comon and Delor (1994) with and without our transformation rules in Section 5 has to be done yet in order to get a precise idea of the benefit from these additional rules.

Finally, also many related questions concerning the complexity of equational formulae have been left out here, e.g. how does the NP-membership result in Section 5 relate to the non-elementary complexity of equational formulae with arbitrary quantifier prefix (cf. Vorobyov, 1996)? What happens to the complexity, when restrictions different from the ones imposed here are considered? In particular, restricting the number of variables rather than restricting the quantifier prefix to $\exists^* \forall^*$, etc.

Acknowledgement

I am very grateful to the anonymous referees for their comments which helped to significantly improve the presentation of this work.

Appendix. The transformation rules of Comon and Lescanne (1989)

In this Appendix we give an overview of the transformation rules of Comon and Lescanne (1989). Those rules of Comon and Lescanne (1989) which play no role in our argumentation and which are not referenced in this work will be omitted. Moreover, for the exact restrictions on the applicability of the rules and for the rule application strategy required in the satisfiability test for equational problems, Comon and Lescanne (1989) has to be referred to.

Elimination of trivial equations and disequations: T

(T_1) $t = t \to \top$
(T_2) $t \neq t \to \bot$.

Replacement: R

(R_1) $z = t \wedge P \to z = t \wedge P(z \leftarrow t)$
(R_2) $z \neq t \vee P \to z \neq t \vee P(z \leftarrow t)$.

Merging: M

(M_1) $s = t \wedge s = u \to s = t \wedge t = u$
(M_2) $s \neq t \vee s \neq u \to s \neq t \vee t \neq u$
(M_3) $s = t \wedge s \neq u \to s = t \wedge t \neq u$
(M_3) $s = t \vee s \neq u \to t = u \vee s \neq u$.

Universality of Parameters: U

(U_1) $(\forall\vec{y})[P \wedge y \neq t] \rightarrow \bot$ if $y \in \vec{y}$
(U_2) $(\forall\vec{y})[P \wedge (y \neq t \vee R)] \rightarrow (\forall\vec{y})[P \wedge R(y \leftarrow t)]$ if the following conditions hold:

1. $y \in \vec{y}$,
2. $y \notin \mathrm{Var}(t)$.

Cleaning: CR

(CR_1) $(\exists w)P \rightarrow P$ if $w \notin \mathrm{Var}(P)$
(CR_2) $\exists(\vec{w}, w)(w = t \wedge P) \rightarrow (\exists\vec{w})P$ if $w \notin \mathrm{Var}(P, t)$.

Clash: C

(C_1) $f(t_1, \ldots, t_m) = g(u_1, \ldots, u_n) \rightarrow \bot$ if $f \neq g$
(C_2) $f(t_1, \ldots, t_m) \neq g(u_1, \ldots, u_n) \rightarrow \top$ if $f \neq g$.

Decomposition: D

(D_1) $f(t_1, \ldots, t_m) = f(u_1, \ldots, u_m) \rightarrow t_1 = u_1 \wedge \cdots \wedge t_m = u_m$
(D_2) $f(t_1, \ldots, t_m) \neq f(u_1, \ldots, u_m) \rightarrow t_1 \neq u_1 \vee \cdots \vee t_m \neq u_m$.

Occur Check: O

(O_1) $z = t \rightarrow \bot$ if $z \in \mathrm{Var}(t)$ and $z \not\equiv t$
(O_2) $z \neq t \rightarrow \top$ if $z \in \mathrm{Var}(t)$ and $z \not\equiv t$.

Universality of Parameters: U'

(U_3) $(\forall\vec{y})[P \wedge z = t] \rightarrow \bot$
if the following conditions hold:

1. $z \not\equiv t$,
2. the equation $z = t$ contains at least one parameter from $\vec{y}$.

The following rule is only correct in the case of an infinite domain:

(U_4) $(\forall\vec{y})[P \wedge (z_1 = u_1 \vee \cdots \vee z_n = u_n \vee R)] \rightarrow (\forall\vec{y})[P \wedge R]$
if the following conditions hold:

1. Every z_i is a variable syntactically different from u_i,
2. every equation $z_i = u_i$ contains at least one parameter from $\vec{y}$,
3. R contains no parameter from $\vec{y}$.

The following rule can only be applied in the case of a finite domain:

(U_5) $(\forall\vec{y})[P \wedge Q] \rightarrow (\forall\vec{y})[P \wedge Q(y \leftarrow a_1) \wedge \cdots \wedge Q(y \leftarrow a_K)]$ if the domain D is of the form $D = \{a_1, \ldots, a_K\}$.

Explosion: E

(E) $(\forall \vec{y})P \to \bigvee_{f \in \Sigma} (\exists \vec{w})(\forall \vec{y})[P \wedge s = f(w_1, \ldots, w_{\alpha(f)})]$
if the following conditions hold:

1. Each f is a (constant or function) symbol from the signature Σ with arity $\alpha(f) \geq 0$,
2. the w_i's are fresh, pairwise distinct variables,
3. s is an argument of an equation or disequation in P and s contains no parameter.

References

Abiteboul, S., Hull, R., Vianu, V., 1995. Foundations of Databases, Addison-Wesley Publishing Company.

Baader, F., Siekmann, J.H., 1994. Unification theory. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A. (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Oxford University Press, pp. 41–125.

Caferra, R., Peltier, N., 1995. Extending semantic resolution via automated model building: applications. In: IJCAI' 95, Proc. 14th Int. Joint Conf. on Artificial Intelligence, Morgan Kaufmann, Montreal, Canada, pp. 328–334.

Caferra, R., Zabel, N., 1991. Extending resolution for model construction. In: van Eijck, J. (Ed.), JELIA' 90, Proc. Logics in AI, European Workshop, Lecture Notes in Artificial Intelligence, no. 478. Springer-Verlag, Amsterdam, The Netherlands, pp. 153–169.

Colmerauer, A., 1984. Equations and inequations on finite and infinite trees. In: FGCS' 84, Proc. Int. Conf. on Fifth Generation Computer Systems, OHMSHA Ltd. Tokyo and North-Holland, Tokyo, Japan, pp. 85–99.

Comon, H., Delor, C., 1994. Equational formulae with membership constraints. Information and Computation 112 (2), 167–216.

Comon, H., Lescanne, P., 1989. Equational problems and disunification. Journal of Symbolic Computation 7 (3/4), 371–425.

Fermüller, Ch., Leitsch, A., 1996. Hyperresolution and automated model building. Journal of Logic and Computation 6 (2), 173–203.

Gottlob, G., Pichler, R., 1999. Working with ARMs: complexity results on atomic representations of Herbrand models. In: LICS' 99, Proc. 14th Annual IEEE Symp. on Logic in Computer Science, IEEE Computer Society, Trento, Italy, pp. 306–315 (full version to appear in Information and Computation).

Kunen, K., 1987. Answer sets and negation as failure. In: Lassez, J.-L. (Ed.), ICLP' 87, Proc. 4th Int. Conf. on Logic Programming, MIT Press, Melbourne, Victoria, Australia, pp. 219–228.

Lassez, J.-L., Maher, M., Marriott, K., 1991. Elimination of negation in term algebras. In: Tarlecki, A. (Ed.), MFCS' 91, Proc. 16th Int. Symp. on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, no. 520. Springer-Verlag, Kazimierz Dolny, Poland, pp. 1–16.

Lassez, J.-L., Marriott, K., 1987. Explicit representation of terms defined by counter examples. Journal of Automated Reasoning 3 (3), 301–317.

Lugiez, D., 1989. A deduction procedure for first order programs. In: Levi, G., Martelli, M. (Eds.), ICLP' 89, Proc. 6th Int. Conf. on Logic Programming, MIT Press, Lisbon, Portugal, pp. 585–599.

Maher, M., 1988. Complete axiomatizations of the algebras of finite, rational and infinite trees. In: LICS' 88, Proc. 3rd Annual Symp. on Logic in Computer Science, IEEE Computer Society, Edinburgh, Scotland, UK, pp. 348–357.

Martelli, A., Montanari, U., 1982. An efficient unification algorithm. ACM Transactions on Programming Languages and Systems 4 (2), 258–282.

Pichler, R., 1999. Solving equational problems efficiently. In: Ganzinger, H. (Ed.), CADE-16, Proc. 16th Int. Conf. on Automated Deduction, Lecture Notes in Artificial Intelligence, no. 1632. Springer-Verlag, Trento, Italy, pp. 97–111.

Pichler, R., 2000a. On the complexity of equational problems in CNF over a finite domain. In: Baumgartner, P., Zhang, H. (eds), FTP 2000, Proc. 3rd Int. Workshop on First-Order Theorem Proving. Fachberichte Informatik der Universität Koblenz-Landau, St. Andrews, Scotland, UK, pp. 182–193. Available at http://www.uni-koblenz.de/fb4/publikationen/gelbereihe/RR-5-2000/.

Pichler, R., 2000b. Speeding up algorithms on atomic representations of Herbrand models via new redundancy criteria. Journal of Symbolic Computation 29 (2), 213–257.

Robinson, J.A., 1965. A machine oriented logic based on the resolution principle. Journal of the ACM 12 (1), 23–41.

Sato, T., Motoyoshi, F., 1991. A complete top-down interpreter for first order programs. In: Saraswat, V.A., Ueda, K. (Eds.), ILPS' 91, Proc. Int. Symp. on Logic Programming, MIT Press, San Diego, California, USA, pp. 35–53.

Stockmeyer, L.J., 1976. The polynomial time hierarchy. Theoretical Computer Science 3 (1–12).

Ullman, J.D., 1989. Principles of Database and Knowledge Base Systems, no. II. Computer Science Press, Rockville, MD, USA.

Vorobyov, S., 1996. An improved lower bound for the elementary theories of trees. In: McRobbie, M.A., Slaney, K. (Eds.), CADE-13, Proc. 13th Int. Conf. on Automated Deduction, Lecture Notes in Artificial Intelligence, no. 690. Springer-Verlag, New Brunswick, NJ, USA, pp. 316–327.